UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

SIMULACIÓN DEL COMPORTAMIENTO DE LA SANGRE EN ARTERIOLAS USANDO UN MODELO DE REJILLA DE BOLTZMANN CON FRONTERAS ABIERTAS

T E S I S

QUE PARA OPTAR POR EL GRADO DE :

DOCTOR EN INGENIERÍA

INGENIERÍA MECÁNICA – MECÁNICA APLICADA

P R E S E N T A :

MC. MARCOS MATÍLDE JESÚS FLORES

TUTOR:

DRA. SUEMI RODRÍGUEZ ROMO

2006

A mi familia
Mónica, Lizbeth, Socorro y Mariela

A mi asesor, Dra. Suemi Rodríguez Romo
gracias por su amistad y por compartir su experiencia y conocimiento

Agradecimientos

Un especial agradecimiento a mi esposa por su compresión y paciencia en estos años de dedicación. Agradezco también por su amistad e incondicional apoyo a Oscar Ibáñez Orozco y José Luis Velásquez Ortega. Y a mis amigos Roberto Díaz y Gilberto Amaya Ventura.

Un agradecimiento muy especial con admiración y respeto a mi asesor : Dra. Suemi Rodríguez Romo por guiarme correctamente en momentos difíciles.

Quiero también dar un especial agradecimiento a los miembros de mi comité tutelar : Dr. Jorge Carrera Bolaños y Dr. Marcelo López Parra por su amable atención y el valioso tiempo que me dedicaron.

Indice Temático

APÉNDICES

OBJETIVOS Y ALCANCES

Objetivo Principal.

Investigar, desarrollar y validar con datos experimentales un modelo de la circulación sanguínea que coadyuve a una mejor comprensión del comportamiento de la sangre en arteriolas, mediante la simulación de factores que afectan la viscosidad de la sangre , usando como herramienta una rejilla bidimensional de Boltzmann.

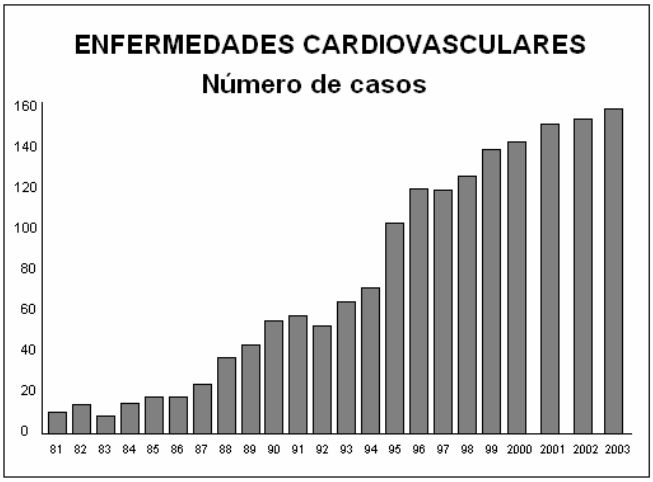
Alcance.

- Simular el comportamiento de la sangre como un fluido de la potencia.
- Proponer un nuevo mecanismo de simulación de flujo con fronteras abiertas en rejillas de Boltzmann para la modelación del flujo sanguíneo.
- Crear un modelo de arteriola basado en información real que permita simular en forma parcial el comportamiento de la sangre en su interior, obteniendo datos importantes como lo son : perfiles de velocidad, presión, velocidad de corte (shear rate) y esfuerzo de corte (shear stress).
- Validar el modelo con datos experimentales reales publicados por el Laboratorio de Investigación en Ingeniería Nuclear de la Universidad de Tokio en Japón.

INTRODUCCIÓN

IMPORTANCIA DEL ESTUDIO DEL COMPORTAMIENTO DE LA SANGRE

Como una consecuencia de la vida moderna las enfermedades cardiovasculares son una de la principales causas de muerte en el mundo, las cuales en los últimos años han registrado un dramático aumento [1]. (ver gráfica A)



Gráfica A. Incremento en el número de enfermedades cardiovasculares.

Estas enfermedades consisten, entre otras, en una anomalía del flujo sanguíneo que con el tiempo y en forma gradual afecta el funcionamiento de diversos órganos internos lo cual puede conducir a la muerte. Entre las enfermedades con un mayor índice de incidencia se encuentran [2,3] :

ENFERMEDAD	DESCRIPCIÓN
Trombosis	Aparición de coágulos en la sangre.
Leucemia	Sobreproducción de glóbulos blancos inmaduros y atípicos.
Hipotensión Arterial	Baja presión arterial.
Infarto	Muerte de alguna porción del corazón.
Apoplejía	Falta de irrigación de la sangre en el cerebro.
Soplo Cardiaco	Sonido anormal que causa la sangre al paso por el corazón.

ENFERMEDAD	DESCRIPCIÓN
Anemia	Disminución del número de glóbulos rojos en la sangre .
Hipertensión Arterial	Alta presión arterial.
Aneurisma	Dilatación localizada en una arteria o vena.
Artereosclerosis	Oclusión de los vasos sanguíneos.
Policitémia	Aumento de los glóbulos rojos, glóbulos blancos y plaquetas.
Trombositopénia	Disminución del número de plaquetas en la sangre.

El desarrollo de diversas técnicas y métodos de diagnóstico, análisis, monitoreo y predicción, así como su diseminación en diversos hospitales, permite que estas enfermedades sean mejor comprendidas y combatidas. Lo que ha conducido a crear adelantos tecnológicos en cardiología, entre los que destacan dispositivos de asistencia circulatoria, técnicas intervensionistas e implantes cardiovasculares como lo son el corazón artificial y la sangre artificial [3].

Todo esto ha llevado a la necesidad de comprender los aspectos biomecánicos implícitos en la fisiología y fisiopatología del sistema cardiocirculatorio, especialmente a lo que concierne a la dinámica del flujo sanguíneo.

Para poder comprender a detalle el comportamiento de la circulación sanguínea en los seres vivos se han desarrollado numerosas técnicas de análisis experimental entre las que destacan por su eficacia, por un lado, las técnicas invasoras o directas y por otro las técnicas no invasoras o indirectas [4,5].

TÉCNICAS INVASORAS O DIRECTAS
Angiografía .- Es una prueba diagnostica que utiliza un colorante intravenoso para valorar la circulación de la sangre en las arterias y/o venas.
Catéteres con sensado bipolar .- Son sensores que se introducen vía intravenosa con el fin de medir perfiles de velocidad sanguínea, presión arterial y concentración de partículas suspendidas en la sangre.

TÉCNICAS NO INVASORAS O INDIRECTAS
Microscopia óptica.- Uso de microscopios de alta resolución.
Microscopia Fluorescente.- Por ingeniería genética, se puede lograr que una célula fabrique una molécula compuesta por una proteína de interés unida a una proteína fluorescente, a la cual se le analiza su trayectoria por medio de microscopía óptica.
Ecografía (**Ultrasonido Doppler**)**.-** La ecografía Doppler es una técnica basada en ultrasonido que permite estudiar de una manera no invasiva, en tiempo real y de forma indirecta la vascularización de los órganos.
Tomografía de rayos X.- La tomografía computada es un análisis de rayos x utilizado para diagnósticos. Las radiografías se toman desde una serie de ángulos diferentes y se acomodan por medio de una computadora para ver un corte seccional de los órganos en el cuerpo.
Tomografía de resonancia magnética nuclear.- Es una técnica de diagnóstico por imágenes que utiliza el magnetismo de los núcleos atómicos de la materia (tejidos, órganos, huesos, etc.) captado por microimanes. Esta señal es traducida en imágenes muy precisas, las cuales revelan un sin número de situaciones anómalas que un profesional experimentado puede interpretar para el diagnóstico de diversas patologías.

Estas técnicas han proporcionado conocimiento e información relevante que ayuda a comprender de mejor manera como la circulación sanguínea proporciona nutrientes a los tejidos, conduce hormonas de una parte del cuerpo a otra y mantener en funcionamiento óptimo todas las células que componen al cuerpo humano [6].

SIMULACIÓN COMPUTACIONAL

Conceptos Básicos

Sistema : Un sistema por definición es un conjunto de objetos o elementos reunidos con alguna interdependencia, que interactúan entre sí como consecuencia de dicha dependencia para lograr un fin común [7,8].

Se define como **Entidad** a un objeto de interés en un sistema. Llamamos **Atributo** a la propiedad de una entidad. Una entidad puede tener varios atributos. **Actividad** es todo proceso que provoca cambios en el sistema. **Estado del sistema** es la descripción de las entidades, atributos y actividades en algún instante de tiempo. El progreso del sistema se estudia siguiendo los cambios en el tiempo-espacio del estado de sistema [8].

Clasificación de sistemas: Existen básicamente dos tipos de sistemas, los sistemas discretos y los continuos. En los discretos el estado del sistema cambia sólo en ciertos puntos en el tiempo. Para ejemplificar, una fábrica es un ejemplo de sistema discreto pues los cambios ocurren en instantes específicos (el pedido de la materia prima o el empaquetamiento). En un sistema continuo, el estado del sistema cambia a cada instante. Se podría comparar con el movimiento de un vehículo cuyos atributos son la posición, la velocidad, la aceleración, etc., que cambian continuamente [9].

Un sistema discreto además puede clasificarse en [9] :

Sistema de terminación: Es aquel en el que existen puntos de inicio y terminación precisos y conocidos.

Sistema de no terminación: Es aquel que está en funcionamiento y carece de puntos de inicio y terminación precisos y conocidos.

Definición de Modelo

Para tratar de predecir cómo se comportará un sistema antes de ser construido es necesario hacer un modelo del sistema. Se define como ‘Modelo’ al esquema teórico de un sistema o realidad compleja que se elabora para facilitar su compresión y estudio.

No existe un único modelo para un sistema, pues éste depende de la naturaleza de la información que se reúne. La tarea de obtener un modelo de un sistema se dividirá en forma genérica en dos partes: la primera es determinar la estructura del modelo y la segunda es proporcionar los datos requeridos. La determinación de la estructura fija la frontera del sistema e identifica las entidades, atributos y

actividades del sistema. Los datos suministran los valores de los atributos que pueden tener y definen las relaciones involucradas en las actividades. La tarea de crear una estructura y la tarea de suministrar los datos se definen como partes de una misma tarea más que como dos tareas por separado, debido a que por lo general están tan íntimamente relacionadas que no se puede hacer una sin la otra [10,11].

Tipos de modelos: Se han utilizado muchos tipos de modelos en los estudios de sistemas, además de haberse clasificado en una diversidad de maneras.

Se consideran principalmente a los modelos como **modelos físicos** y **modelos matemáticos**. Ambos se definen, de manera general, como una formulación o una ecuación que expresa las características esenciales de un sistema físico o de un proceso en términos matemáticos. Por ejemplo, a través de sus observaciones, Newton formuló su segunda ley del movimiento, la cual establece que la razón de cambio del momento con respecto al tiempo de un cuerpo, es igual a la fuerza resultante que actúa sobre él. La expresión matemática, o el modelo, de la segunda ley es la ya conocida ecuación : F =ma, donde 'F' es la fuerza neta que actúa sobre el objeto, 'm' es la masa del objeto y 'a' es su aceleración [10,11].

Definición de Simulación por Computadora

Al tratar algunos problemas del mundo real, el modelo resultante o prototipo puede ser tan complejo o grande que no es posible o práctico desarrollar, como un ejemplo se encuentra la construcción de naves espaciales en donde millones de partes interactúan entre si y el diseño detallado de mecanismos juega un papel importante para el buen funcionamiento de la nave. El no contar con un sistema de simulación por computadora se tendrían que construir de antemano los millones de partes antes de saber si éstas funcionaran correctamente [7].

En la simulación por computadora, como el término indica, se diseña y construye un modelo por computadora que imita el argumento real del problema. Entonces se usa el modelo para aprender cómo se comporta el sistema, formulándose preguntas del tipo : "¿qué sucedería si... ? " [7].

VENTAJAS DE LA SIMULACIÓN POR COMPUTADORA

1.- Como con cualquier forma de simulación, la simulación por computadora permite que las personas experimenten con muchas políticas y argumentos diferentes sin cambiar o experimentar realmente con el sistema real.

2.- La simulación permite analizar problemas complejos para los que no están disponibles resultados analíticos. De hecho, la mayoría de los problemas de mundo real encajan en esta categoría. La simulación proporciona una alternativa práctica.

3.- La simulación por computadora permite reducir tiempo. Por ejemplo, se puede estudiar el impacto a largo plazo de una política para un banco durante todo un año en una simulación por computadora que dura unos cuantos minutos. La alternativa de implementar realmente la política y observar sus resultados en un año no es práctica.

IMPORTANCIA DE LA SIMULACIÓN COMPUTACIONAL EN EL ESTUDIO DEL COMPORTAMIENTO DE LA SANGRE

En las dos últimas décadas, investigadores de diversas áreas tales como ingeniería, medicina y biología comenzaron a introducir técnicas predictivas dentro de la práctica de la medicina [1,5]. El actual grado de desarrollo alcanzado por las técnicas de modelado computacional aliado a la potencia de cálculo de las computadores, ha permitido el estudio y desarrollo de diversos modelos computacionales capaces de anticipar, con aceptable grado de precisión, los resultados de importantes procedimientos médicos. Tal es el caso, del desarrollo de las técnicas de reconstrucción de imágenes tridimensionales que a partir de una secuencia de informaciones bidimensionales proporcionadas por aparatos tales como CT (tomografía computarizada) y MR (resonancia magnética), permiten construir modelos computacionales para un paciente específico. Estos modelos al proporcionar visualizaciones tridimensionales de alta resolución que representan los fenómenos que están ocurriendo en una determinada parte del organismo, pueden contribuir al establecimiento de mejores procedimientos médicos.

Por otro lado, el creciente índice de mortalidad por enfermedades cardiovasculares también ha motivado a los investigadores en la creación de diversos modelos computacionales que ayuden a un mejor entendimiento y compresión del funcionamiento del sistema cardiovascular humano. Tal es el caso, que en los últimos años diversos investigadores [1,4] han presentado modelos tridimensionales que muestran el comportamiento del flujo de la sangre en válvulas cardiacas artificiales y en segmentos de vasos sanguíneos.

BIOMECÁNICA CARDIOCIRCULATORIA

El enfoque biomecánico ofrece una visión más detallada del sistema cardiocirculatorio (SCC), aportando los conceptos necesarios para aprovechar los desarrollos tecnológicos, y plantear modelos para realizar simulaciones, que permiten evaluar el comportamiento del sistema bajo diferentes condiciones fisiológicas y patológicas, como alternativa a complejos estudios en el sistema vivo, sea por dificultad técnica, por el riesgo sobre el paciente o por el alto costo [4].

La sangre fluye a través del sistema cardiocirculatorio (SCC) gracias al impulso que genera el corazón con cada contracción. Desde este punto de vista, se puede comparar este sistema biológico con un sistema de bombeo mecánico. En ambos casos, el comportamiento del sistema se puede analizar descomponiéndolo en tres subsistemas: el fluido (la sangre) que es el medio de transporte de masa, momento y calor; la red de conductos (vasos sanguíneos), a través de los cuales se desplaza el fluido; y la unidad de bombeo (el corazón), que provee la energía necesaria para impulsar el fluido y vencer las pérdidas de presión [1].

Es difícil elaborar un modelo completo del SCC, por un lado por las muchas variables que intervienen y por otro por las complejas relaciones entre sus componentes que llevan a modelos demasiado complejos, difíciles de resolver, y cuyos resultados son difíciles de interpretar. De las observaciones experimentales del SCC se pueden fijar algunas condiciones que simplifican el tratamiento de cada subsistema y facilitan la solución de los modelos [3].

La sangre es una solución acuosa que contiene diversas partículas en suspensión y exhibe una viscosidad que depende de varios factores; esto dificulta la descripción de su comportamiento y la elaboración de modelos que describan los fenómenos de flujo [1].

Los vasos sanguíneos son conductos formados por varias capas, las cuales están compuestas por diversos materiales, conformando estructuras que se caracterizan por una respuesta elástica compleja ante la aplicación de un esfuerzo. Se pueden crear modelos que evalúen en forma aproximada la respuesta elástica de los vasos ante esfuerzos externos (de los tejidos que lo rodean) e internos (de la presión sanguínea). Asimismo la interacción entre la sangre (presión sanguínea) y el vaso (tensión de pared) se puede evaluar con relaciones mecánicas, como lo han mostrado las observaciones experimentales [3].

CAPÍTULO I

Rejillas de Boltzmann

1.1.- IMPORTANCIA DEL TEMA.

El Estudio del Movimiento de los Fluidos :

El movimiento de los fluidos puede tener distintos comportamientos. Un comportamiento simple como puede ser el caso de un flujo laminar en un tubo, o un comportamiento complejo, como puede ser el movimiento de un flujo en forma de torbellino en un cilindro. Muchas de estas situaciones han sido estudiadas experimentalmente, sin embargo, resulta ventajoso desarrollar modelos numéricos capaces de simular muchos de los flujos experimentados en el movimiento de diferentes fluidos.

El movimiento de un fluido es gobernado por la ecuación de continuidad (ver apéndice A) :

$$\frac{d\rho}{dt} = -\rho\left(\nabla \cdot v\right) \tag{1.1}$$

y la ecuación de Navier-Stokes (ver apéndice A) :

$$\rho \frac{dv}{dt} = -\nabla p + \mu \nabla^2 v + \rho g \tag{1.2}$$

La ecuación de Navier-Stokes es una ecuación diferencial la cual no se conoce la solución analítica excepto para un número pequeño de casos especiales. Con el advenimiento de la tecnología computacional se han realizado simulaciones numéricas de flujo de fluidos.

Métodos Numéricos en el Estudio del Comportamiento de los Fluidos :

En el área de la dinámica de fluidos computacional (DFC) se han desarrollado numerosas técnicas para resolver la ecuación de Navier-Stokes (1.2) y la ecuación de continuidad (1.1) o una ecuación derivada de ellas. Otro desarrollo que hasta el momento ha sido menos popular es el desarrollo de la dinámica molecular.

- Soluciones Numéricas de la Ecuación de Navier-Stokes :

  El método más popular en (DFC) es la solución numérica de la ecuación de Navier-Stokes (1.2). Dada la ecuación de Navier-Stokes y un conjunto de condiciones de frontera adecuado es posible resolverla en una rejilla usando una técnica numérica estándar. Esto resultará bien para flujos simples, sin embargo, para problemas más complejos frecuentemente se requiere un planteamiento más complejo. Existen muchos textos bibliográficos referentes a este tema incluyendo a Roach [12] y Conner and Brebbia [13].

- Dinámica Molecular .

Una forma para simular el comportamiento de un fluido en una computadora es modelar el comportamiento individual de las moléculas. Para esto el sistema debe representar las interacciones intermoleculares de forma correcta de tal forma que su comportamiento sea como el de un fluido real [14]. Entonces situaciones diferentes del flujo pueden ser modeladas cambiando la energía promedio de las moléculas y la separación entre ellas.

La desventaja principal con este planteamiento es la gran cantidad de recursos computacionales que son requeridos. Ahora bien el sistema debe ser capaz de calcular y actualizar con nuevos valores y en pocos pasos de tiempo la nueva posición y velocidad de todas las partículas en cuestión tomando en cuenta las interacciones con las demás partículas vecinas. Cualquier partícula que colisione debe ser también identificada por el sistema en cada paso de tiempo y calcular su nueva trayectoria. Todo esto puede consumir bastante tiempo de cómputo [15] por lo que el sistema puede tomar horas en encontrar una posible solución al movimiento del fluido, aun cuando el número de moléculas sea relativamente pequeño como puede ser el caso de un gas como fluido.

- Modelando una Rejilla de Gas.

En los pasados 15 años, nuevos métodos han sido desarrollados para la simulación computacional de fluidos: el método de retícula de gas, es uno de los primeros. En lugar de considerar un gran número de moléculas individuales, como es el caso del acercamiento de la dinámica molecular, un número pequeño de partículas de fluido son consideradas. Una partícula virtual de fluido es un grupo grande de moléculas, que aunque es más grande que una molécula, se considera aun pequeña en la escala macroscópica. Esto reduce la cantidad de datos que necesitan ser almacenados después de grandes simulaciones, las cuales pueden ser desarrolladas usando menos de un millón de estas partículas virtuales [16].

El uso de partículas virtuales se justifica en terrenos en los que las propiedades macroscópicas no dependen directamente del comportamiento microscópico del fluido. O bien, las propiedades en una escala no dependen de los detalles de la escala inmediata anterior, en lo general [17]. En el modelo de retícula de gas se restringe a las partículas virtuales a moverse a través de una retícula fundamental y el movimiento se desarrolla en lapsos discretos de tiempo denominadas iteraciones. Las leyes de conservación se incorporan en las reglas que se aplican a cada nodo de la red en cada iteración. Es importante señalar que las leyes en que se fundamenta este modelo son las leyes de la mecánica clásica. Un autómata celular desarrollado para modelar el comportamiento microscópico tiene la ventaja de que puede ser implementado eficientemente y

en forma rápida en una computadora. Las propiedades del fluido son incorporadas en las reglas de actualización.

- El Modelo de Rejilla de Boltzmann .

El modelo de rejilla de Boltzmann evolucionó a partir del modelo de retícula de gas, debido al gran número de dificultades que se encontraron en el modelo de retícula de gas para simular correctamente el comportamiento de fluidos, entre ellas se encuentra invarianza rotacional y ruido estadístico. Estas dificultades fueron claramente superadas y varios cambios fueron hechos dando como resultado el modelo de rejilla de Boltzmann, donde, como su nombre lo indica, involucra la simulación de la ecuación de Boltzmann.

$$\frac{\partial f}{\partial t} + v \cdot \frac{\partial f}{\partial x} + F \cdot \frac{\partial f}{\partial v} = \Omega (f) \tag{1.3}$$

donde $\Omega(f)$ es una función de colisión, $F(x,t)$ es el la fuerza del cuerpo por unidad de masa, $v(x,t)$ es la velocidad de la partícula y $f(x,v,t)$ es la función de distribución de probabilidad. La función de distribución de probabilidad es una función probabilística con el cual las propiedades macroscópicas del fluido pueden ser encontradas. La simulación de la ecuación de Boltzmann es llevada a cabo en una rejilla y la forma de la función de colisión Ω puede ser tomada del operador de colisión BGK [16]. El modelo es procesado de la misma manera como lo es el modelo de retícula de gas excepto que ahora, en vez de considerar partículas individuales viajando a lo largo de las ligas de la retícula, es la función de distribución la cual es procesada.

El modelo de rejilla de Boltzmann tiene ventajas asociadas las cuales el modelo de retícula de gas no pudo superar. Por esta razón el modelo de rejilla de Boltzmann es una herramienta más funcional para la simulación del comportamiento de fluidos.

1.2.- QUE SON LOS AUTOMATAS CELULARES.

El modelo de retícula de gas tiene su origen en los autómatas celulares (AC), es por esto que para comprender el principio funcional del modelo de rejillas de Boltzmann es necesario entender la mecánica de funcionamiento de los AC.

La modelación de la mayoría de los sistemas en ingeniería está basada en modelos matemáticos, los cuales representan teóricamente la naturaleza del fenómeno. Generalmente, para modelar algunos sistemas macroscópicos se usan ecuaciones diferenciales. Cuando no se puede usar una aproximación en continuo del problema, algunos procedimientos de discretización y digitalización de sistemas, permiten realizar análisis numéricos sobre modelos discretos.

Un método matemático común utilizado para modelar algunos sistemas mecánicos es el método de elementos finitos el cual involucra una discretización y un subsiguiente análisis numérico para comprender el comportamiento de diversos sistemas en ingeniería. No obstante la complejidad de aplicar el método de elementos finitos sobre algunos sistemas reales es tal, que resulta difícil lograr modelos que describan con precisión sus comportamientos [18].

Los autómatas celulares son una opción para construir modelos discretos aproximados de algunos sistemas complejos de naturaleza continua. Incluso es posible, por ejemplo, lograr modelos discretos que representen con suma fidelidad algunas leyes de la física.

La Estructura de un Autómata Celular :

Los autómatas celulares son sistemas dinámicos discretos que involucran cuatro elementos de construcción [18]:

1. El espacio celular
2. Los estados de las células
3. La configuración de vecindades
4. La regla de evolución local

El espacio celular :

> Los autómatas celulares están definidos por un arreglo de células de dimensión ' d ', llamado espacio celular; donde cada uno de los elementos del arreglo se denomina *célula*, que es un término tomado de las ciencias biológicas como algunos otros tales como ``evolución'', ``vida'', ``muerte''; que son adecuados para comprender el comportamiento individual de los elementos.

El espacio celular, en principio y tal como fue definido originalmente, es infinito en todas sus direcciones, y posteriormente debido principalmente a las limitaciones prácticas de cómputo, el espacio celular puede estar acotado.

Para poder simular el espacio celular infinito, se ha decidido tomar condiciones periódicas de frontera, esto es, que la columna de células siguiente a la última es la primera, y la columna de células anterior a la primera es la última. Cuando d=1 el espacio celular es un anillo, en d=2 se forma un espacio toroidal.

El conjunto de estados de las células

El estado de cada célula es el valor que tendrá ésta en el siguiente paso del tiempo y está determinado por el estado actual de ella misma y de las células vecinas más cercanas en un radio de vecindad predeterminado.

Un autómata celular lineal está definido por el par (k,r), donde ‘k’ es el número de estados para cada célula y ‘r’ es el radio de vecindad. Los autómatas celulares que se usaran como muestra a continuación son los acl (2,1) que significa: ``autómata celular lineal de dos estados y radio de vecindad de uno''. Un ejemplo de ello se muestra en la figura 1.1.

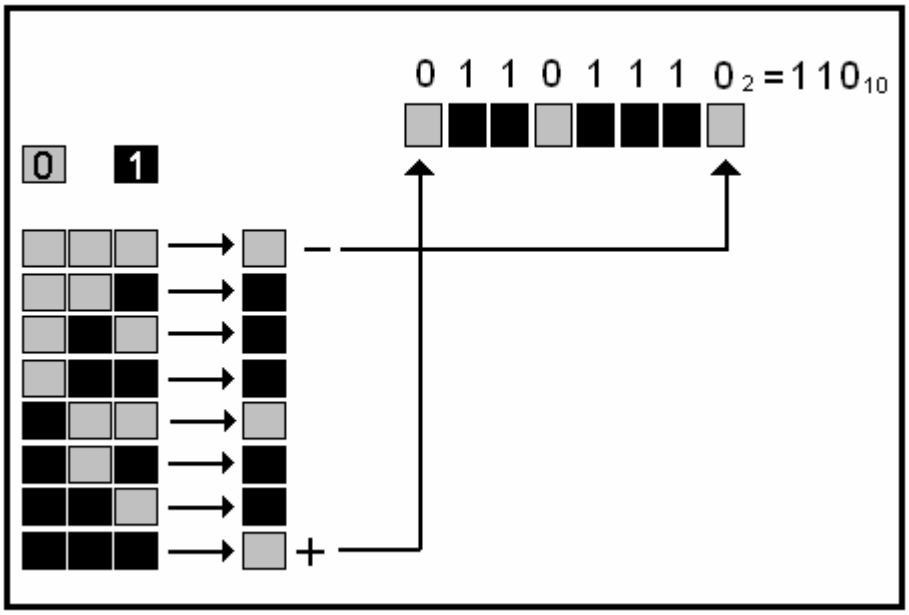


Figura 1.1 : Configuración de vecindades en un acl (2,1).

La configuración de vecindades :

La manera en que las células actúan dentro del espacio celular, está determinada por cómo son afectadas por el entorno que las rodea. A este entorno se le llama vecindad.

Así, la vecindad de una célula, llamada célula en transición, o célula central x_c, es el conjunto de células cercanas a ella a una distancia $d(x_c,x_i) \leq r$, además de la propia célula central. Para este caso, la vecindad es formada por el conjunto:

$$V = \{x_i \mid d(x_c, x_i) = 1\} \cup \{x_c\}.$$

En un espacio bidimensional el número de vecinos queda definido por : 2r + 1, siendo r el radio de vecindad.

Si estamos en un espacio de dimensión 1 o unimensional y radio 1, se toma como vecinos a la celda de la izquierda y al de la derecha. Por tanto se tendría la siguiente vecindad:

L – C – R

donde L significa 'vecino de la izquierda', R 'vecino de la derecha' y 'C' celda central. Pero si se esta en un espacio de dimensión 1 y de radio 2, se tendría la vecindad siguiente :

L – L – C – R – R

La regla de evolución local :

Cada célula del espacio celular, toma su valor en dependencia de la configuración de su vecindad. De esta forma, las reglas de evolución local para un acl (2,1) pueden ser definidas por :

L	C	R	Valor
0	0	0	ϵ_0
0	0	1	ϵ_1
0	1	0	ϵ_2
0	1	1	ϵ_3
1	0	0	ϵ_4
1	0	1	ϵ_5
1	1	0	ϵ_6
1	1	1	ϵ_7

Donde los valores de C_i son valores arbitrarios fijados de antemano para una regla en particular. Por ejemplo para un caso particular, los siguientes valores de vecindad se relacionan con la secuencia binaria 01101110 que genera la regla de evaluación local 110_{10} (ver también figura 1.1).

L	C	R	Valor
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

De aquí se deduce que para un acl (2,1) el número de posibles configuraciones para las vecindades es k^{2r+1} y el número de posibles reglas de evolución queda definido por $k^{k^{2r+1}}$. Por ejemplo, si k=2 y r=1 . el número de posibles configuraciones para las vecindades es 8 y el número de posibles reglas de evolución es 256.

El problema de las reglas de evolución, radica en el número exponencial de posibles reglas. El caso anterior es uno de los casos más sencillos, cuando se aumenta el número de estados, en un acl (3,1) habrá entonces 3^3 configuraciones diferentes para las vecindades, y 3^{27} posibles reglas de evolución. Y si en lugar de aumentar el número de estados, se aumenta el tamaño de la vecindad, entonces el número de posibles reglas es 2^5 posibilidades para las vecindades y 2^{32} reglas diferentes.

El manejo del tiempo en la evolución de los autómatas celulares es muy simple, cada paso en el tiempo es contado progresivamente con números naturales, reservando el cero para la configuración global inicial, como se aprecia en la figura 1.2.

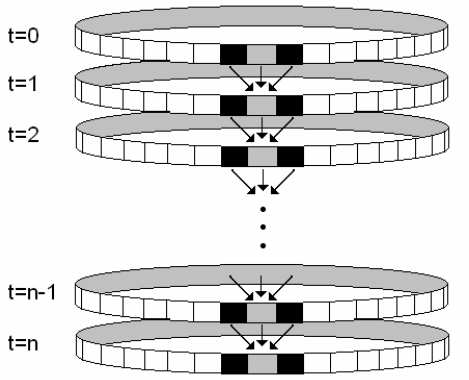


Figura 1.2 : La función de evolución global de un autómata celular determina la configuración global en el siguiente paso de tiempo en la evolución del autómata celular.

1.3.- TEORÍA CINÉTICA DE LOS GASES .

Dinámica Atómica :

Consideremos una colección de n moléculas moviéndose en una caja de volumen V a una temperatura T y mutuamente interactuando vía potencial intermolecular $\Phi(\vec{r})$, siendo $\vec{r}$ la separación intermolecular. Si el tamaño lineal s de las moléculas, el cual es básicamente el rango efectivo del potencial de interacción, es tan pequeño que su separación intermolecular promedio es $d = (V/N)^{1/3}$, las moléculas pueden tratarse aproximadamente como si fueran partículas sin estructura, de tal forma que su dinámica viene siendo gobernada por las ecuaciones clásicas de Newton :

$$\frac{d\vec{x}_i}{dt} = \frac{\vec{p}_i}{m}, \tag{1.4}$$

$$\frac{d\vec{p}_i}{dt} = \vec{F}_i, \qquad i = 1.....,N, \tag{1.5}$$

donde $\vec{x}_i$ es la coordenada de posición de la i-enésima partícula, $\vec{p}_i \equiv m\vec{v}_i$ su momento lineal y $\vec{F}_i$ la fuerza experimentada por la i-enésima partícula como un resultado de las interacciones intermoleculares y sus campos externos posibles (gravedad, campo eléctrico etc.). Se puede pensar que la solución de las ecuaciones (1.4) y (1.5) puede ser en principio simple, pero esto es inviable debido a dos principales razones : La primera es que N es generalmente del orden del número de Abogador (10^{23}), lo que lo hace imposible de manejar en cualquier computador. Y segundo, auque esto fuera posible de hacer, es tanta la información que el tiempo requerido para llegar a algún posible resultado sería utópico.

Afortunadamente en términos de física observable nosotros estamos interesados en el comportamiento de variables macroscópicas tales como presión del fluido y temperatura, originadas de un promedio estadístico sobre la historia de un gran número de partículas individuales, entendiéndose como promedio estadístico a un promedio espacial sobre un volumen termodinámico el cual viene siendo una región del espacio suficientemente pequeña con respecto a las dimensiones globales del dominio macroscópico y lo suficientemente grande como para contener una muestra de moléculas con significado estadístico. Un ejemplo típico lo es la densidad del aire, la cual en condiciones normales, es del orden de $n=2.687\times10^{19}$ moléculas/cm^3 (número de Loschmidt), de lo que podemos inferir que un milímetro cúbico de aire almacena alrededor de 10^{16} moléculas, con un error estadístico de diez partes por billón. Este mismo análisis puede ser hecho en varios niveles de complejidad.

A un nivel de cinética de un cuerpo la pregunta principal es : ¿ Cual es la probabilidad de encontrar una molécula alrededor de una posición $\vec{x}$ en un tiempo t con un momento $\vec{p}$?

La función de densidad de probabilidad $f(\vec{x},\vec{p},t)$ y la cantidad $\Delta n = f\Delta\vec{x}\Delta\vec{p}$, donde $\Delta\vec{x},\Delta\vec{p}$ son puntos cúbicos finitos centrados alrededor de $\vec{x}$ y $\vec{p}$ en el así llamado espacio-face $\mu = [(\vec{x},\vec{p}) : \vec{x},\vec{p} \in R^3]$. En 1872, el científico austriaco Ludwing Boltzmann derivo una ecuación describiendo la evolución de f en términos de interacciones microdinámicas. Esta es la celebrada ecuación de Boltzmann (BE), uno de las más grandes logros en física teórica hechos en el siglo antepasado.

La ecuación cinética para la función de distribución de un distribución de un cuerpo se lee de la siguiente forma :

$$D_t f = [\partial_t + \frac{\vec{p}}{m}\cdot\partial_{\vec{x}} + \vec{F}\cdot\partial_{\vec{p}}]f(\vec{x},\vec{p},t) = C_{12} \qquad (1.6)$$

donde el término del lado izquierdo representa el movimiento del flujo de moléculas a lo largo de las trayectorias asociadas con el campo de fuerza $\vec{F}$ y C_{12} representa el efecto de las colisiones intermoleculares tomando las moléculas que entran y que salen de la trayectoria del flujo.

El operador de colisión involucra a la función de distribución f_{12} de dos cuerpos, expresando la probabilidad de encontrar una molécula, por decir 1, alrededor de $\vec{x}_1$ con velocidad $\vec{v}_1$ y una segunda molécula, por decir 2, alrededor de $\vec{x}_2$ con velocidad $\vec{v}_2$, ambas en un tiempo t .

En la ecuación (1.6) Boltzmann hizo algunas suposiciones en términos de la naturaleza del sistema físico : Se trata de un gas diluido con moléculas sin estructura e interactuando vía un potencial entre dos cuerpos de rango corto. Bajo tales condiciones, las interacciones intermoleculares pueden ser descritas únicamente en términos de colisiones binarias localizadas, con moléculas viajando en trayectorias libres e independientes unas de otras. Bajo esta suposición el operador de colisión en términos de ganancias (G) y perdidas (L) se puede expresar como [17,18,19]:

$$C_{12} \equiv G - P = \int (f_{1'2'} - f_{12})\, g\sigma(g,\Omega)\, d\Omega\, d\vec{p}_2 \qquad (1.7)$$

correspondiendo a las colisiones directas e inversas tomando moléculas de afuera y de adentro del diferencial de volumen $d\vec{V}_1$, respectivamente. Expresando el número de moléculas con velocidad relativa $\vec{g} = g\vec{\Omega}$ alrededor del ángulo sólido $\vec{\Omega}$ (ver figura 1.3).

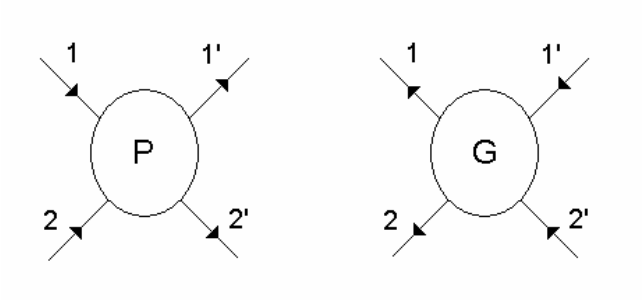


Figura 1.3. Diagrama simbólico de colisiones directas e inversas.

Relajación al Equilibrio Local :

El término equilibrio local describe la situción en la cual las cantidades termodinámicas del sistema tales como densidad, temperatura, presión, etc. pueden variar espacialmente con el tiempo, pero en cada elemento de volumen individual las relaciones termodinámicas entre los valores que aplican localmente se siguen obedeciendo [20].

$$f_1^{'}\, f_2^{'} = f_1\, f_2 \tag{1.8}$$

A la ecuación (3.8) se le conoce como condición de balance detallado y significa que cualquier colisión directa / inversa es dinámicamente balanceada por una pareja inversa / directa.

$$\ln(f_1^{'}) + \ln(f_2^{'}) = \ln(f_1) + \ln(f_2) \tag{1.9}$$

Los logaritmos de los componentes de la ecuación (1.9) nos muestra que ln f es un invariante de colisión aditivo; que es una propiedad microscópica, la cual no cambia bajo el efecto de las colisiones. La consecuencia inmediata es que se tiene un equilibrio termodinámico. ln f debe ser una función solo de los invariantes de colisión $I(v) = [1, m\vec{v}, mv^2/2]$ (número, momento, conservación de energía). Esto lleva a que :

$$\ln f = A + B_a v_a + \frac{1}{2} C v^2 \tag{1.10}$$

donde A, B_a, C son tres multiplicadores de Lagrange que llevan a cabo la dependencia funcional en el conjugado de los campos hidrodinámicos ρ, u_a, E, llamados normalmente densidad, momento y energía. Estos parámetros de Lagrange son calculados al imponer la conservación de estas cantidades.

$$m \int f \, d\vec{v} = \rho \tag{1.11}$$

$$m \int f \, v_a \, d\vec{v} = \rho \, v_a \tag{1.12}$$

$$m \int f \, \frac{v^2}{2} \, d\vec{v} = \rho \, e \tag{1.13}$$

donde ρ es la densidad del fluido (masa por unidad de volumen) , u_a es la velocidad macroscópica del flujo y ρe la densidad de la energía.

Las Ecuaciones de Navier-Stokes :

Basándose en el procedimiento de Chapman-Enskog para recuperar las ecuaciones de Navier-Stokes [21,22], se obtienen las siguientes equivalencias para un fluido isotérmico:

$$\partial_t \rho + \partial_a \rho \, u_a = 0 \tag{1.14}$$

$$\partial_t \rho \, u_a + \partial_b \, T_{ab} = \partial_b \tau_{ab} \tag{1.15}$$

donde ρ es la densidad del fluido y u_a el campo de flujo. Las cantidades tensoriales son definidas de la siguiente forma de acuerdo al mismo procedimiento de Chapman-Enskog :

$$T_{ab} \equiv \rho \, u_a u_b + P \delta_{ab} \ , \tag{1.16}$$

$$\tau_{ab} \equiv \mu [\partial_a u_b + \partial_b u_a - 2/3 \, (\partial_c u_c) \delta_{ab}] \tag{1.17}$$

donde *P* es la presión del fluido y μ es la viscosidad dinámica.

El tensor T_{ab} es un espejo de la dinámica Newtoniana no disipativa, mientras que τ_{ab} representa el efecto disipativo asociado con la relajación local al equilibrio. De hecho, los componentes en equilibrio y no equilibrio de la función de distribución, pueden mostrar que estos dos tensores representan correspondientemente los componentes de equilibrio y no equilibrio del tensor P_{ab} respectivamente :

$$T_{ab} = m\int f^{eq} v_a v_b d\vec{v} \tag{1.18}$$

$$\tau_{ab} = m\int f^{neq} v_a v_b d\vec{v} \tag{1.19}$$

La física del flujo del fluido es controlada principalmente por dos parámetros adimensionales, el numero de Mach y el número de Reynolds :

$$Ma = \frac{u}{c_S} \tag{1.20}$$

$$\mathrm{Re} = \frac{uL}{v} \tag{1.21}$$

donde c_S es la velocidad del sonido y $v = \mu/\rho$ es la viscosidad cinemática. El número de Mach mide la fuerza relativa de los términos inerciales $\partial_b u_a u_b$ (fluido llevado alrededor de su propio momento) contra gradientes de presión, mientras que el número de Reynolds es una medida de inercia contra efectos disipativos.

El Modelo de Ecuación Bhatnagar-Gross-Krook :

Es importante mencionar que para facilitar la solución numérica y analítica de la ecuación de Boltzmann, el operador de colisión debe ser simplificado [23] asumiendo que la función de distribución de probabilidad de las partículas relaja a su estado de equilibrio a una velocidad constante. De esta forma, la complicada integral no lineal del operador de colisión es remplazada por una expresión más simple con el fin de eliminar la dificultad matemática sin estropear la física básica. Esta simplificación de las ecuaciones de Boltzmann es llevada a cabo mediante el llamado operador de colisión BGK (Bhatnagar-Gross-Krook) [24] :

$$C_{BGK}(f) = -\frac{f - f^{eq}}{\tau} \tag{1.22}$$

Aquí $f \equiv f_1$ y f^{eq} es un equilibrio local parametrizado por una conservación local de densidad ρ, velocidad $\vec{u}$ y temperatura T, mientras que τ es una escala típica de tiempo asociada con la relajación al equilibrio local.

En principio, el tiempo de relajación τ es un parámetro complicado de la función de distribución de probabilidad f. La drástica simplificación asociada con BGK es la suposición de un valor constante para esta escala de relajación. La principal simplificación asociada con la ecuación BGK es que la no localidad en espacio momento del operador de colisión es canalizada dentro de una dependencia funcional en las variables hidrodinámicas conservadas, las cuales sirven como parámetros de orden para la función de distribución de la partícula.

1.4.- AUTÓMATA CELULAR DE REJILLA DE GAS .

Stainslas Ulam y Jhon von Neuman introdujeron el concepto de autómata celular (AC) en los pasados años 1950's. Ulam y Neuman propusieron usar computadoras analógicas simples para llevar acabo sus modelos, en las cuales el procesamiento de la información se hacia de una manera analógica no digital. Sus ideas fueron construir redes de computadoras donde cada computadora fuera un nodo de una rejilla física. Como estas computadoras estaban conectadas en una red, cada estado de una computadora (nodo) era determinado por el estado de sus vecinos. Este sistema de computadoras interconectadas se conoce hoy en día como autómata celular.

El primer esfuerzo para simular un sistema de flujo de fluidos vino en 1973 por Hardi, Pomeau and Pazzis [17], ellos introdujeron el autómata celular de retícula de gas (Lattice Gas Cellular Automaton (LGCA)). El nombre de su modelo fue HPP, derivado de las iniciales de sus nombres. Ellos usaron las reglas de un autómata celular para lograr que la masa y el momento se conservaran. Si bien en el modelo HPP se conservaban la masa y el momento, a nivel macroscópico no se podía recuperar las ecuaciones de Navier-Stokes. En 1986, Frisch, Hasslancher y Pomeau descubrieron que las ecuaciones de Navier-Stokes podrían ser recobradas usando un AC sobre una rejilla hexagonal.

Su modelo fue conocido como el modelo FHP, que es derivado del nombre de sus autores. Después introdujeron al modelo FHP, diferentes versiones del AC, donde propusieron modelar fluidos en forma tridimensional (d'Humieres,1986).

1.4.1.- EL MODELO HPP .

El modelo HPP es un modelo binario simple en el sentido computacional , donde el fluido es considerado como una colección de partículas virtuales, las cuales pueden ser un conglomerado de átomos o moléculas. Dichas partículas residen en los vértices de una rejilla cuadrada, como es mostrado en la figura 1.4 .

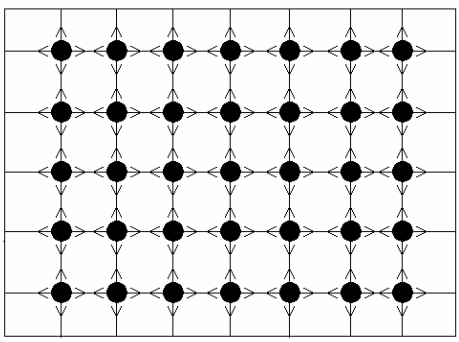


Figura 1.4. Retícula cuadrada usada en el modelo HPP.

Todas las partículas tienen masa positiva arbitraria y velocidad unitaria. En cada vértice existen cuatro posibles direcciones de velocidad para cada partícula virtual del fluido. Una variable booleana es usada para indicar la presencia o ausencia de una partícula de fluido en cada sitio de la rejilla en una dirección dada. Ver figura 1.5.

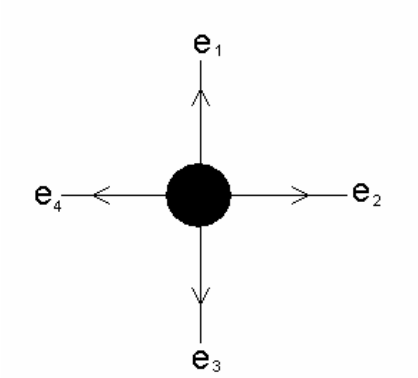


Figura 1.5. Direcciones de velocidad.

Pueden llegar a existir hasta cuatro partículas en un solo sitio de la rejilla en un momento dado, pero a no más de una se le permite viajar en cada dirección. El movimiento de la partícula es actualizado en dos pasos. Primero, cada partícula se mueve hacia un vecino próximo en una dirección específica de velocidad. Segundo, cualquiera de dos partículas que colisionan hacia el frente, se mueven en ángulo recto hacia su dirección de origen. La colisión es gobernada por la conservación de masa y la conservación de momento, Ver figura 1.6.

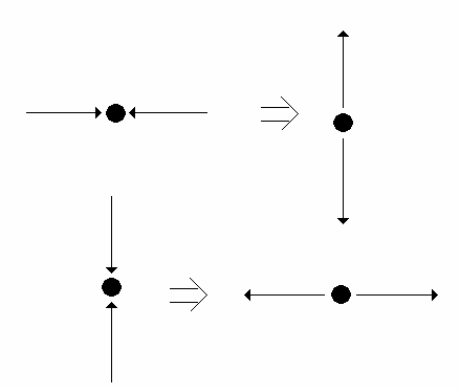


Figura 1.6. Reglas de colisión para el modelo HPP.

La figura 1.7 muestra el proceso de colisión en detalle en diferentes lapsos de tiempo .

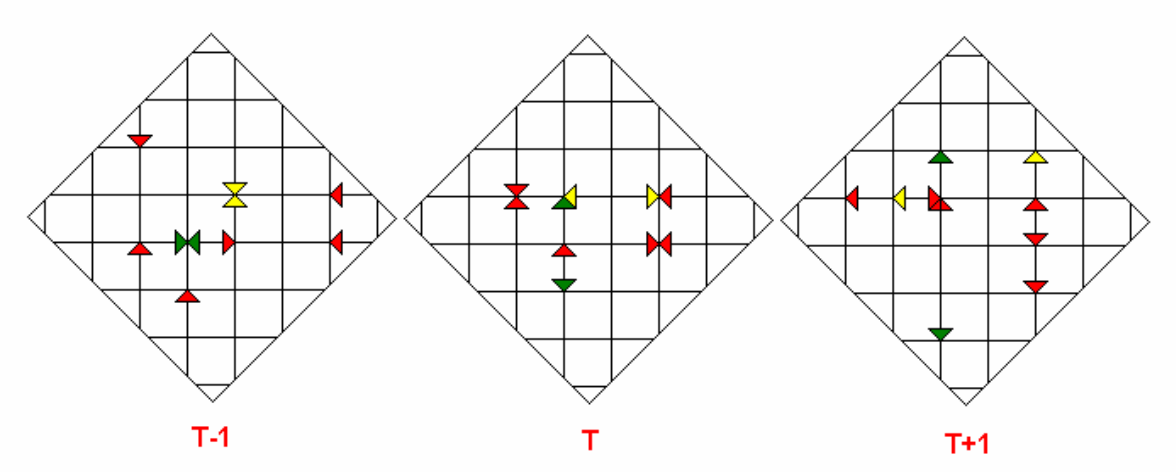


Figura 1.7. Proceso de colisión entre partículas virtuales de fluido al aplicar reglas de colisión y viéndolo en diferentes lapsos de tiempo (T-1,T,T+1).

El modelo HPP es absolutamente estable; sin embargo, sufre de una falta de isotropía, además de tener la característica de que el modelo no recupera las ecuaciones de Naver-Stokes a nivel macroscópico, debido a un inadecuado grado de simetría rotacional en la rejilla cuadrada.

1.4.2.- EL MODELO FHP .

En 1986, Frisch, Hasslacher y Pomeau introducen el modelo FHP. Su principal enfoque fue resolver el problema de isotropía que fue hallado en el modelo HPP. Ellos introdujeron el modelo de autómata celular en una rejilla hexagonal, donde partículas virtuales con masa unitaria y velocidad unitaria se mueven en los vértices de una rejilla hexagonal discreta. Ver figura 1.8.

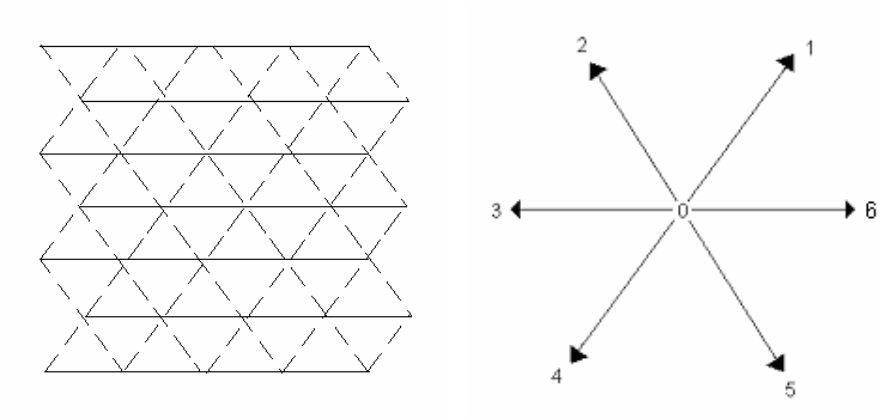


Figura 1.8. (a) Rejilla hexagonal , (b) Direcciones de velocidad.

Existen seis posibles direcciones de velocidad en cada sitio de la rejilla. Las direcciones de velocidad difieren 60 grados entre ellas, tal como se muestra en la figura 1.7. Pueden existir hasta seis partículas en cada sitio de la rejilla, pero no más de una es permitida viajar en la misma dirección. Una variables booleana es usada en cada dirección de la rejilla para indicar la presencia o ausencia de una partícula en esa dirección.

El método de actualización es similar al del modelo HPP. Las partículas son actualizadas en cada colisión y en cada iteración. Las colisiones de las partículas en una rejilla hexagonal son determinadas por una prescripción de reglas de colisión que contienen todos los posibles estados de colisión. Básicamente, todas las posibilidades de colisión están compuestas de dos y tres configuraciones de colisión. Pueden existir hasta 64 tipos de colisión diferentes, pero debido a la simetría, el número de colisiones puede ser reducido a 16. Los 16 posibles resultados son gobernados por la conservación de masa y la conservación de momento. El flujo de partículas es llevado acabo por el movimiento de las partículas a una distancia unitaria de la red en la dirección de sus velocidades. Cantidades macroscópicas tales como masa y momento pueden ser recobradas por el modelo FHP usando promedios estadísticos.

La figura 1.9 muestra las reglas de colisión que son manejadas en el modelo FHP.

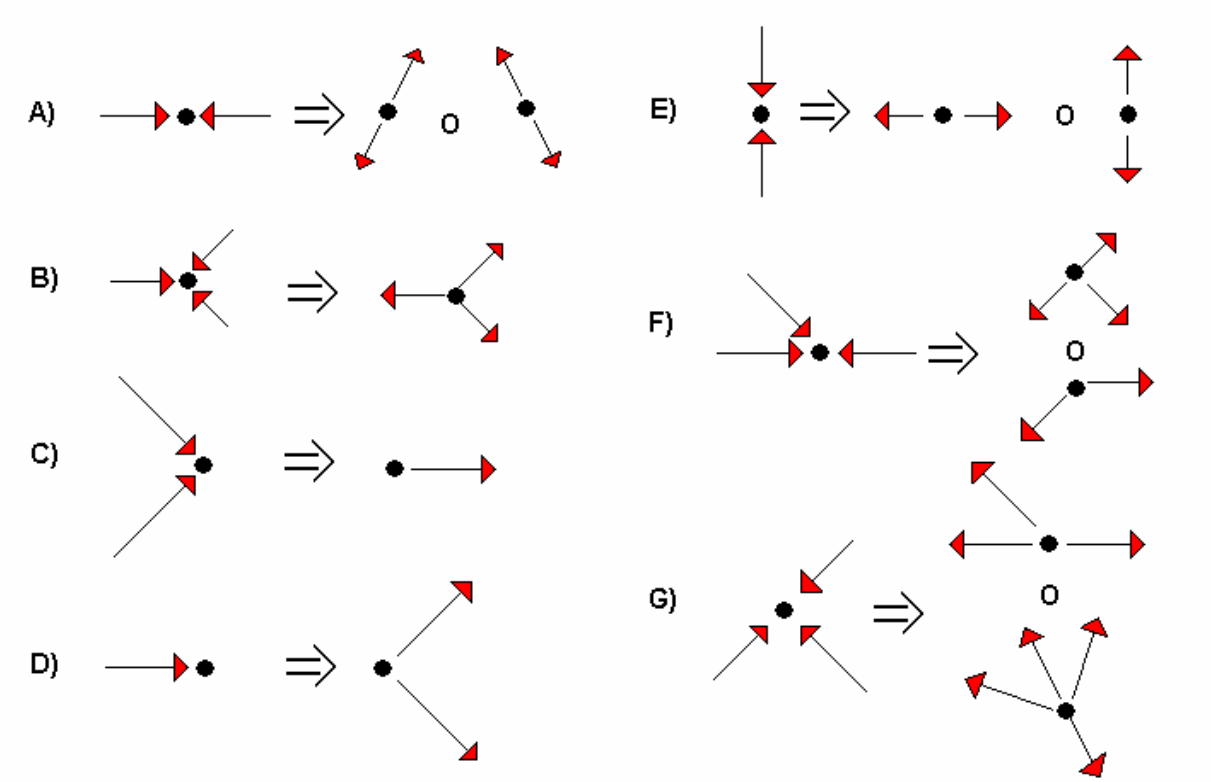


Figura 1.9. Reglas de colisión en el modelo FHP.

Este modelo también sufre algunos inconvenientes que los hacen un modelo deficiente en la simulación de fluidos reales. Esta son : el ruido estadístico por el estado binario de las partículas y una complejidad excesiva si se desea crear modelos en 3D.

1.5.- MODELOS DE REJILLA DE BOLTZMANN CON MICRODINÁMICA BOOLEANA SUBYACENTE .

- **LBE No Lineal :**

El primer modelo de LBE fué propuesto por G. McNamara y G. Zanetti en 1988, con la idea explicita de eliminar el problema de ruido estadístico que presentaba su antecesor LGCA [18]. La idea básica era simple : solo reemplazar los números de ocupación booleana n_i por poblaciones ensamble-promedio correspondientes.

$$f_i = < n_i > \ , \tag{1.23}$$

Esto es conveniente para separar el promedio y la parte de fluctuación de los números de ocupación booleana :

$$n_i = f_i + g_i \tag{1.24}$$

donde el promedio de fluctuaciones g_i tiende a cero por definición.

integrando la ecuación (1.24) en la ecuación de microdinámica booleana :

$$n_i \ (\vec{x} + \vec{C}_i, t+1) = n_i (\vec{x}, t) \tag{1.25}$$

se obtiene :

$$\Delta_i f_i = C_i \ (f) + G_i \tag{1.26}$$

donde G_i colecta todas las contribuciones de las correlaciones interparticulares. Si se supone que no existe ninguna correlación entre partículas cuando colisionan, se tiene que :

$$G_i = 0 \tag{1.27}$$

Y eliminando todas las correlaciones multipartícula se obtiene una ecuación de diferencias finitas no lineal para la función de distribución de una partícula f_i :

$$\Delta_i f_i = C_i \ (f_1, ..., f_2) \tag{1.28}$$

Esta es la primera ecuación de rejillas d Boltzmann .

Ahora se puede ver que la no linealidad de LBE es debida a una transcripción directa de la microdinámica del LGCA y que el ruido estadístico es eliminado porque f_i es por definición un promedio.

- **La Cuasilinealidad de LBE :**

El problema de la inviabilidad de la no linealidad de LBE para simulaciones en tres dimensiones fue resuelto por Higuera y Jiménez al proponer una forma cuasilineal de LBE :

$$\Delta_i f_i = A_{ij} (f_j - f_j^{eq}) \tag{1.29}$$

donde la notación A_{ij} es conocida como matriz de dispersión y su relación exacta es :

$$A_{ij} = \sum_{ss'} (s'_i - s_i)\, A(s,s')(s'_j - s_j)\, d^P (1-d)^q \tag{1.30}$$

donde $p = \sum_i^b s_i$ es el número de partículas en el estado s y q es su complemento para b, q = b – p +1 , conocido con el nombre de número de huecos.

- **La Matriz de Dispersión A_{ij} :**

Como un ejemplo se toma a la matriz de dispersión de 6x6 del modelo FHP. Existen solo cuatro posibles ángulos de dispersión : θ = [0, π/6, π/3, π], a los cuales se asocian 4 elementos independientes en la matriz de dispersión: a=0, b=π/6, c=π/3, d=π. Con esta notación, la matriz de dispersión para el modelo FHP es explícitamente escrita como:

$$A_{i,j} = \begin{bmatrix} a & b & c & d & c & b \\ b & a & b & c & d & c \\ c & b & a & b & c & d \\ d & c & b & a & b & c \\ c & d & c & b & a & b \\ b & c & d & c & b & a \end{bmatrix} \tag{1.31}$$

Donde los requerimientos de conservación de masa y momento son impuestos de antemano para asegurar que se cumplan las leyes de la mecánica, mediante las siguientes reglas :

$$\sum_i A_{ij} = 0\,, \qquad j = 1,\ldots,b \tag{1.32}$$

$$\sum_i c_{ia}\, A_{ij} = 0\,, \qquad j = 1,\ldots,b \qquad a = 1,\ldots,d \tag{1.33}$$

1.6.- REJILLA BHATNAGAR – GROSS – KROOK .

➢ Relajación en una Sola Escala de Tiempo .

Como se mostró en el apartado anterior, la matriz de dispersión y el equilibrio local pueden ser considerados como parámetros libres de la teoría. Bajo este concepto, la viscosidad de los fluidos en LB modelo D2Q9 es controlada en su totalidad por un solo parámetro identificado como el eigenvalor principal distinto de cero de la matriz de dispersión A_{ij}. Los eigenvalores restantes son entonces usados para minimizar la interferencia de campos no hidrodinámicos (fantasmas) con la dinámicas macroscópica observable. Esto conduce a hacerse naturalmente una pregunta : Puesto que el transporte esta relacionado con el manejo de un solo eigenvalor distinto de cero, porque no simplificar las cosas en un futuro seleccionando adecuadamente un parámetro de la matriz de dispersión?. Este punto fue analizado por diversos investigadores [19], llegando a la siguiente conclusión: Tal propuesta es equivalente a seleccionar la matriz de dispersión en una forma diagonal si se asume que la función de distribución de probabilidad de las partículas f relaja a su estado de equilibrio a una velocidad constante :

$$A_{i\,j} \rightarrow \omega\, \delta_{i\,j} \tag{1.34}$$

donde el parámetro $\omega > 0$ es el inverso del tiempo de relajación para el equilibrio local. Esto significa moverse de un esquema de relajación de varios tiempos a un esquema de relajación de un solo tiempo, en el cual todos los modos de relajan en la misma escala de tiempo $\tau = 1/\omega$.

De acuerdo a esto, la diagonal de LBE se lee como sigue :

$$\Delta_i f_i = -\omega\,(f_j - f_j^{eq}) \tag{1.35}$$

dando la conexión directa con el modelo Bhatnagar – Gross – Krook de la ecuación de Boltzmann en teoría cinética continua [20]. La ecuación (1.35) ha sido apropiadamente llamada Rejilla BGK o LBGK en forma corta, por sus siglas en ingles.

Es difícil imaginar una forma más simple de esta ecuación para recobrar las ecuaciones de Navier-Stokes. Aunque aún algunos puntos importantes necesitan ser especificados.

Como ya se habrá notado, la diagonalización del operador de colisión implica que todos los modos decaen a la misma velocidad. Esto es indeseable para campos fantasmas y absolutamente inadmisible para cantidades conservadas.

Dejando a un lado los fantasmas por un momento, y considerando solo los campos hidrodinámicos. La solución es simple. Puesto que las leyes de conservación de masa y momento no pueden ser codificadas en un cierto nivel de la matriz de dispersión (de ninguna manera una matriz diagonal puede cumplir con todas las reglas de las ecuaciones (1.32) y (1.33)), éstas deben necesariamente ser forzadas en el equilibrio local precisamente como en la teoría cinética continua.

Como una variación al modelo cuasilineal de LBE, se requiere explícitamente un equilibrio local que transmita la misma cantidad de densidad y momento, como la función de distribución actual :

$$\sum_i f_i^{eq} = \sum_i f_i = \rho \tag{1.36}$$

$$\sum_i f_i^{eq} c_{i\,a} = \sum_i f_i c_{i\,a} = \rho\, u_a \tag{1.37}$$

- **El Equilibrio en el Modelo LBGK .**

Una familia genérica de equilibrio de LBGK puede ser expresada como una expansión de Mach de un multi nivel de energía Maxwelliano, el cual se puede expresar como :

$$f_I^{eq} = \rho\omega_I\left(1 + \frac{c_{I\,a}u_a}{c_S^2} + \frac{Q_{I\,a\,b}\,u_a u_b}{2c_S^4}\right) \tag{1.38}$$

Donde I es una abreviatura para (i, j) y los pesos ω_I y la constante c_S (velocidad del sonido) depende de la selección específica de las velocidades discretas c_{Ia} . Los pesos ω_I pueden ser interpretados como números degradados provenientes de otras dimensiones o simplemente como una diferencia de masas de las partículas que se mueven en diferentes direcciones.

De cualquier forma, conservación de masa y momento, como también isotropía, son impuestas por las igualdades siguientes :

$$\sum_I \omega_I = 1, \tag{1.39}$$

$$\sum_I \omega_I\, c_{I\,a} = 0, \tag{1.40}$$

$$\sum_I \omega_I \, c_{I\,a} \, c_{I\,b} = P \, \delta_{a\,b} \, , \tag{1.41}$$

$$\sum_I \omega_I \, Q_{I\,a} = 0 \, , \tag{1.42}$$

$$\sum_I \omega_I \, c_{I\,a} \, Q_{I\,b\,c} = 0 \, , \tag{1.43}$$

$$\sum_I \omega_I \, c_{I\,a} \, c_{I\,b} \, Q_{I\,b\,c} = \rho \, u_a \, u_b \, , \tag{1.44}$$

Estas ecuaciones admiten varias soluciones debido a los múltiples niveles de energía de la rejilla. Qian (1992) proporciono una familia completa de soluciones para un modelo de velocidad ‘m’ en ‘n’ dimensiones : DnQm, y los llamo factores de peso (ver tabla 1.1)

Los ejemplos más populares son D1Q3, D1Q5, D2Q9, cuyos diagramas son mostrados en las figuras 3.10 y 3.11, junto con una tabla sinóptica (tabla 1.1) que muestra sus respectivos factores de peso.

Figura 1.10. Rejillas D1Q3 y D1Q5 .

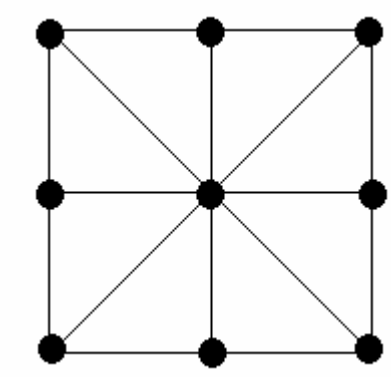


Figura 1.11. Rejillas D2Q9 .

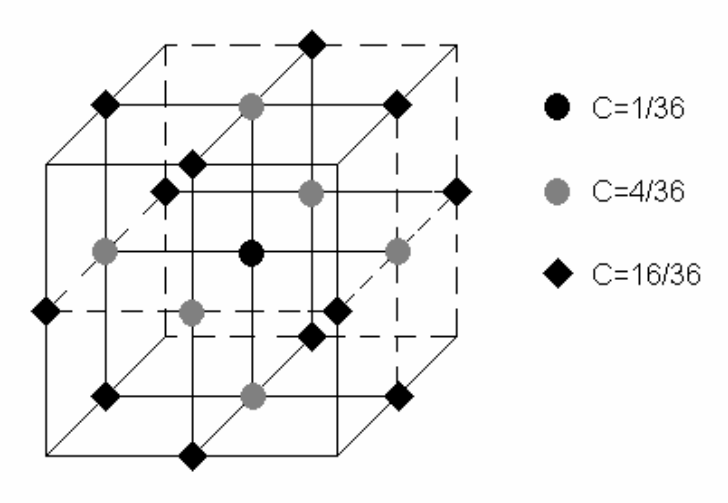


Figura 1.12. Rejillas D3Q19 .

Donde C representa los factores de peso para una rejilla modelo D3Q19. Ver tabla 1.1.

Modelo de rejilla	Factor de Peso
D1Q3	4/6 1/6
D2Q5	6/12 2/12 1/12
D2Q9	16/36 4/36 1/36
D3Q15	16/72 8/72 1/72
D3Q19	12/36 2/36 1/36

Tabla 1.1. Factores de peso para algunos modelos DnQm de rejilla BGK .

- **Comparaciones entre LBGK y LBE .**

La relajación en una sola escala de tiempo, implica que tanto la masa, como el momento y la transferencia de calor, todos ellos, toman lugar en un mismo instante. Esto es apropiado solo para gases ideales.

Pero esta restricción puede ser parcialmente eliminada usando dos operadores de relajación de la forma :

$$C_i = -\omega\,(f_i - f_i^{eq}) - \omega^*(f_{i^*} - f_{i^*}^{eq}) \qquad (1.45)$$

donde i* denota el conjugado de $i\,(\vec{c}_i + \vec{c}_{i^*} = 0)$.

1.7.- TIPOS DE CONDICIONES DE FRONTERA EN REJILLAS DE BOLTZMANN.

La dinámica de los flujos de un fluido es altamente dependiente del medio por el cual circula. Esta influencia es descrita matemáticamente vía la prescripción de condiciones de frontera adecuadas.

Las condiciones de frontera juegan un papel importante puesto que ellas permiten crear soluciones que son compatibles con las limitantes externas. Estas limitantes generalmente hablando, pueden ser quizás triviales bajo ciertas condiciones ideales (flujos periódicos) , pero aun así la selección de una adecuada condición de frontera sigue siendo una tarea delicada.

Se distinguen dos clases básicas de condiciones de frontera : condiciones de frontera para fronteras simples y condiciones de frontera para frontera complejas. Las fronteras elementales pueden ser vistas como situaciones en las cuales la frontera física es alineada con las coordenadas de la rejilla. Las fronteras elementales necesitan ser superficies tersas las cuales formen líneas rectas o planos. La marca distintiva de este tipo de fronteras es que ellas no cortan las celdas de la rejilla.

Las fronteras complejas por el contrario, pueden tomar virtualmente cualquier forma. Consecuentemente su implementación es mucho más difícil, pero tienen un mayor ámbito de aplicación en la ingeniería. Ver figura 1.13.

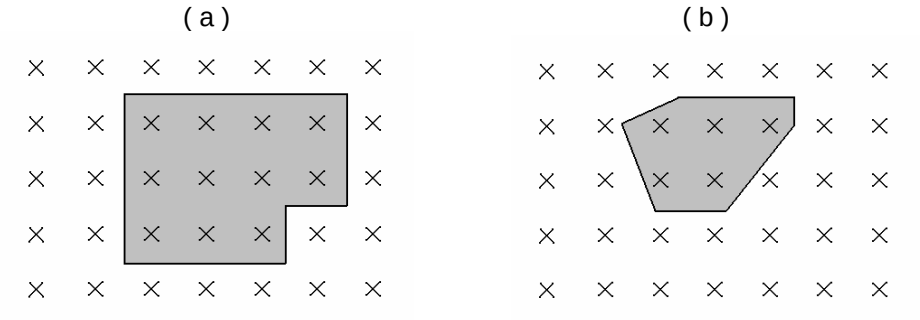


Figura 1.13. (a) Fronteras regulares . (b) Fronteras irregulares .

En el caso de fronteras simples se consideran las siguientes condiciones de frontera :

- Periódicas .
- No deslizantes .
- De deslizamiento libre .
- De deslizamiento con fricción .
- De paredes deslizantes .
- De entradas y salidas abiertas .

- **Condiciones de Frontera Periódicas .**

Las condiciones de frontera periódicas son las más simples de implementar de todas las demás condiciones de frontera. La implementación practica es como sigue [36] : Considerar una rejilla cuadrada consistente de N x M nodos de red. El fluido discreto es representado por 9 matrices $f_i(l,m),\ l=1,...,N,\ m=1,...,M,\ i=0,...,8.$

Se introducen entonces cuatro capas extras (buffers) en los extremos Norte (N), Sur (s), Oeste (O), Este (E). Ver figura 1.14.

$$\mathrm{N}=\left[\, i=0,...,N+1\ ;\ j=M+1\right]$$
$$\mathrm{S}=\left[\, i=0,...,N+1\ ;\ j=0\right]$$
$$\mathrm{O}=\left[\, i=0\ ;\ j=0,...,M+1\right]$$
$$\mathrm{E}=\left[\, i=N+1\ ;\ j=0,...,M+1\right]$$

NO	N	N	N	N	N	N	N	N	N	N	NE
O	f	f	f	f	f	f	f	f	f	f	E
O	f	f	f	f	f	f	f	f	f	f	E
O	f	f	f	f	f	f	f	f	f	f	E
O	f	f	f	f	f	f	f	f	f	f	E
O	f	f	f	f	f	f	f	f	f	f	E
SO	S	S	S	S	S	S	S	S	S	S	SE
0	1	2	3	4	5	6	7	8	9	10	11

Figura 1.14. Rejilla con fronteras periódicas f : fluido, O : frontera oeste, E : frontera este, N : frontera norte, S : frontera sur **.**

La perioricidad a lo largo del eje ‘x’ es impuesta por simplicidad llenando los buffers ‘Oeste’ con las funciones de densidad de la última columna del dominio físico, ver figura 1.17 :

$$\begin{aligned} f_{in,Oeste}(O) &= f_{out,Este}(EF) \\ f_{in,Este}(E) &= f_{out,Oeste}(OF) \end{aligned} \qquad (1.46)$$

Aquí ‘*EF*’ es una abreviatura para el fluido que va hacia el este: $EF=\left[\mathrm{i}=\mathrm{N}\,;\,\mathrm{j}=0,...,\mathrm{N}+1\right].$

El subíndice ‘in’ y ‘out’ denotan poblaciones dirigiéndose hacia adentro y hacia fuera respectivamente. Con el número de velocidad siguiente , ver figura 1.15.

$$\{ in, \quad O\} = \{ 1, \ 2,8 \}$$
$$\{ out, \ O\} = \{ 4,5,6 \}$$
$$\{ in, \quad E\} = \{ out, O \}$$
$$\{ out, \ E\} = \{ in, \quad O \}$$

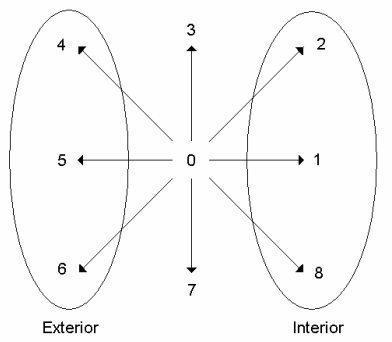


Figura 1.15. poblaciones dirigiéndose hacia adentro y hacia fuera.

Las cuatro esquinas requieren un trato por separado :

$$\mathrm{f}\_\{in\}\{NO\} = \mathrm{f}\_\{out\}\{SE\}$$
$$\mathrm{f}\_\{in\}\{SO\} = \mathrm{f}\_\{out\}\{NE\}$$
$$\mathrm{f}\_\{in\}\{NE\} = \mathrm{f}\_\{out\}\{SO\}$$
$$\mathrm{f}\_\{in\}\{SE\} = \mathrm{f}\_\{out\}\{NO\}$$

ya que ellas son la unión entre las diferentes capas de los extremos.

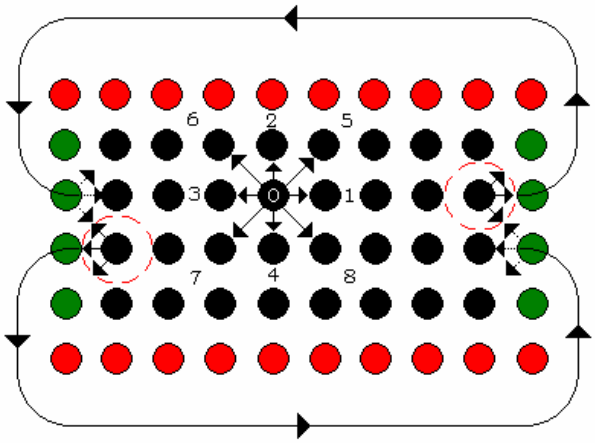


Figura 1.16. Fronteras periódicas: Partículas moviéndose desde los buffers de la derecha hacia los buffers de la izquierda y viceversa.

- **Condiciones de Frontera No Deslizantes .**

La siguiente condición de frontera es también llamada 'bounce-back' y esta basada en el siguiente principio $J_x = J_y = 0$, donde J_x y J_y son las componentes tangencial y normal de la velocidad del fluido respectivamente en las paredes del ducto. Esta condición es físicamente apropiada cada vez que la pared sólida tiene suficiente rugosidad para prevenir cualquier movimiento de fluido neto en la pared. Nuevamente se asume que la superficie sólida esta alineada con los nodos de la rejilla.

Se distinguen dos tipos de implementaciones :

- En el límite de la rejilla y
- En medio de la rejilla .

En el límite de la rejilla significa que la frontera física esta exactamente ubicada en las líneas límite de la rejilla. Y en medio de la rejilla se refiere a la situación donde la frontera se encuentra entre dos líneas de la rejilla. Ver figura 1.18. cabe mencionar que ambos conceptos fueron de gran utilidad y se usaron en el desarrollo de esta tesis.

La situación de el límite de la rejilla es fácil de tratar solo se invierten de lugar todas las poblaciones en un nodo frontera.

$$f_{in}(N) = f_{out}(N) \tag{1.47}$$

$$f_{in}(S) = f_{out}(S) \tag{1.48}$$

Donde N y S denotan norte y sur respectivamente.

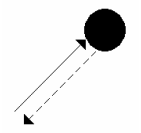


Figura 1.17. Frontera límite de rejilla Bounce-Back en condición de frontera no deslizante .

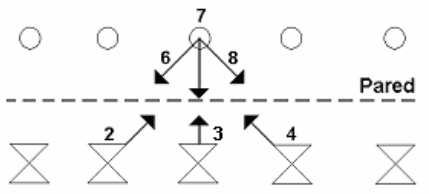


Figura 1.18. Frontera en medio de la rejilla Bounce-Back en condición de frontera no deslizante, las poblaciones 6, 7 y 8 reciben las poblaciones 2, 3 y 4 respectivamente.

Bounce-Back puede ser visto en términos matriciales como una matriz de 3 x 3 :

$$\begin{bmatrix} f_6(x,y) \\ f_7(x,y) \\ f_8(x,y) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} f_2(x,y) \\ f_3(x,y) \\ f_4(x,y) \end{bmatrix}$$

La componente tangencial de la velocidad del fluido y la componente normal desaparecen.

$$J_x = (f_1 + f_2 + f_8) - (f_4 + f_5 + f_6) = (f_1 - f_5) = 0 \qquad (1.49)$$

$$J_y = (f_2 + f_3 + f_4) - (f_6 + f_7 + f_8) = 0 \qquad (1.50)$$

La primer igualdad se asegura inicializando $f_1 = f_5$ en un tiempo t =0. La segunda igualdad es garantizada por la reflexión completa (1.47) y (1.48).

Una relación similar aplica en la versión ‘En medio de la rejilla’, correspondiendo a la transformación :

$$\begin{bmatrix} f_6(x,y) \\ f_7(x,y) \\ f_8(x,y) \end{bmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{bmatrix} f_2(x-1,y-1) \\ f_3(x,y-1) \\ f_4(x+1,y-1) \end{bmatrix}$$

Nuevamente, la pared de abajo es manejada con argumentos simétricos y las esquinas requieren un trato especial.

- **Condiciones de Frontera de Entrada y Salida Abiertas .**

Muchas situaciones en la práctica y en la teoría que son de interés involucran fronteras abiertas, en donde las entradas y salidas son libres. Para flujos abiertos es común asignar un perfil de velocidad dado u_{in} en la entrada, mientras que en la salida se da un valor de velocidad u_{out} determinado.

Considerando primero la condición de la velocidad en la frontera de entrada. Una descripción del flujo de entrada es fácilmente implementada llenando los buffers con la población en equilibrio correspondiente a los valores deseados de densidad y velocidad del flujo.

$$f_{in}(y) = f^{eq}\left[\rho_{in}, u_{in}(y)\right] \tag{1.51}$$

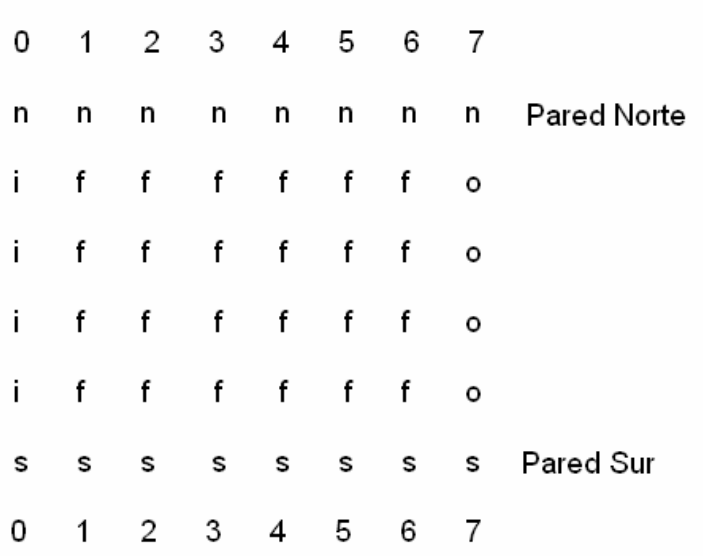


Figura 1.19 : I : Lado de entrada O : Lado de salida.

La salida es más delicada.

Ingenuamente se piensa que una condición de gradiente cero puede ser impuesta simplemente copiando la penúltima columna del canal en la última columna (columna 5 en la columna 6 de la figura 1.19). Desafortunadamente esto no funciona, al menos que el lado de salida se sitúe lo bastante retirado de la corriente de flujo para permitir que el flujo se estabilice hacia un perfil de gradiente cero en una dimensión.

Una forma práctica es aplicar la llamada condición de frontera de ‘tapa porosa’. La idea es que las partículas alcancen la salida y sean reflejadas hacia atrás con una probabilidad ‘r’, la cual es ajustada para asegurar la conservación de masa.

Es usual que se remplace las poblaciones de la salida con sus valores equivalentes de equilibrio (lo cual es equivalente a suponer implícitamente que no existen gradientes en la salida).

El cálculo es el siguiente:

$$J_{out}^{out} = f_1 + f_2 + f_8 = \frac{\rho}{24}(6 + 12u), \tag{1.52}$$

$$J_{in}^{out} = f_5 + f_6 + f_4 = r\frac{\rho}{24}(6 - 12u) \tag{1.53}$$

Donde J_{out}^{out} representa a las poblaciones de salida: f_1, f_2 y f_8 de los nodos de salida de la rejilla. Y J_{in}^{out} representa a las poblaciones de entrada : f_5, f_6 y f_4 de los nodos de salida de la rejilla.

Equiparando la entrada con la salida se tiene que : $J_x = J_{out}^{out} - J_{in}^{out} = \rho u_{in}$
y asumiendo una velocidad transversal cero *v=0* , se obtiene ‘r’ como una función de ‘ u_{in} ’ y de la velocidad u .

$$r = \frac{1 + 2u - 4u_{in}}{1 - 2u} = 1 + 4\left(\frac{u - u_{in}}{1 - 2u}\right) \tag{1.54}$$

Esta relación muestra que si el flujo de salida es exactamente igual a la velocidad de entrada, la reflexión no es requerida (r=1), porque el flujo es automáticamente estable. Nótese también que si $u > u_{in}$, las poblaciones de salida deben ser reflejadas (r<1), para absorber el exceso de velocidad $u - u_{in}$.

1.8.- LBE COMO UN MODELO COMPUTACIONAL PARA LA DINÁMICA DE FLUIDOS.

LBE es una invención sofisticada de mecanismo estadísticos computacionales, este es el humus del cual se desprenden tanto sus ventajas como sus desventajas.

Sin embargo, la pregunta esencial es si LBE representa algo nuevo y no solo es otro esquema de diferencias finitas para las ecuaciones de Navier-Stokes.

Para ver esto con claridad, se explicaran algunas de las características distintivas de LBE dejando aun lado por el momento la teoría cinética y apoyándose en el lenguaje de análisis numérico [25,26].

Las propiedades principales de cualquier esquema numérico pueden ser clasificadas de la siguiente manera [17] :

- Causalidad.
- Precisión.
- Estabilidad.
- Eficiencia.
- Flexibilidad.

Con respecto estas propiedades, LBE puede ser clasificada como una aproximación hiperbólica finita de las ecuaciones de Navier-Stokes con las siguientes características :

- Causalidad : El estado de una variable depende de su estado en un tiempo anterior.
- Precisión : Segundo orden en espacio y tiempo.
- Estabilidad : Estabilidad lineal incondicional y estabilidad no lineal condicional.
- Eficiencia : Buena eficiencia en computadoras seriales y paralelas.
- Flexibilidad : Pobre adaptabilidad en rejillas no uniformes.

➢ Causalidad .

Los esquemas numéricos para ecuaciones dependientes en el tiempo se pueden clasificar en dos categorías, dependiendo de la forma en que la variable tiempo es discretizada. Estas son :

- Explícitos .
- Implícitos .

En esquemas explícitos el presente estado de una variable dada es calculado en términos de un número (usualmente pequeño) de vecinos en un tiempo anterior. Así, el estado actual es único y explícitamente especificado en términos del estado anterior de los vecinos. Por poner el ejemplo de una ecuación en una dimensión, un esquema explicito típico se lee de la siguiente manera :

$$f(x_l, t_n) = \sum_{m,k<n} T_{lm,kn}\ f(x_m, t_k) \tag{1.55}$$

donde los coeficientes $T_{lm,kn}$ son conectores espacio-tiempo entre la localidad actual espacio-tiempo $(x_l = l\Delta x,\ t_n = n\Delta t)$ y sus vecinos.

En esquemas implícitos el estado actual de una variable dada depende del estado simultaneo de sus vecinos :

$$f(x_l, t_n) = \sum_{m,k<n} T_{lm,kn}\ f(x_m, t_k) + \sum_{m} T_{lm,nn}\ f(x_m, t_n) \tag{1.56}$$

El efecto operacional es una perdida de localidad. Para calcular el valor actual de (x_l, t_n), se necesita resolver el sistema lineal (1.56) involucrando todo el espacio de localidades en el tiempo t_n.

LBE pertenece a la familia de esquemas de evolución en el tiempo. Para probar esto considérese la forma diferencial del operador de colisión :

$$D_t f \equiv (\partial_t + u_a \partial_a) f \tag{1.57}$$

y discretizandolo de acuerdo a un esquema lagrangiano de primer orden con la característica $dx_a = u_a dt$.

El resultado es :

$$\Delta_a f \equiv f(\vec{x} + u_a dt, u_a, t + dt) - f(\vec{x}, u_a, t) + O(dt) = 0 \tag{1.58}$$

Ahora haciendo $u_a = c_{ia}$, $i = 1,...,b$ para el conjunto de velocidades discretas se obtiene precisamente el lado izquierdo de LBE con $dt = 1$.

- **Precisión .**

Precisión se refiere a los errores introducidos por el reemplazo de operadores diferenciales por unos de diferencias finitas. Si este error es cero para los polinomios de grado *p*, entonces se dice que la discretización tiene una veracidad de *p*-enésimo orden.

Por ejemplo, una diferencia parcial finita de la forma $(f(x+\Delta x)-f(x))/\Delta x$ reproduce exactamente la derivada de una función lineal, pero no una cuadrática (o de mayor orden) , lo que significa que la veracidad es de primer orden.

Para discutir la veracidad espacial de LBE es conveniente considerar un par de paridades conjugadas de velocidades discretas $\vec{c}_i$ y $\vec{c}_{i*} = -\vec{c}_i$. Ahora se introduce la suma de los operadores de colisión $\Delta_i^+ \equiv \Delta_i + \Delta_i^*$ asociada con el movimiento libre a lo largo de la dirección i-enésima.

Esto se lee como (omitiendo el índice i) $\Delta^+ f = (f(x+\Delta x) - 2f(x) + f(x-\Delta x))/2\Delta x$, donde $\Delta x = c\Delta t$. Esto comprueba que es una discretización centrada de la derivada de segundo orden de $\partial_{XX} f$, la cual se sabe que es de segundo orden de veracidad.

- **Estabilidad .**

La noción básica es que la rejilla es un mundo discreto el cual puede solo soportar señales con una velocidad de propagación finita. El criterio necesario para el concepto de estabilidad es simplemente que la información física no debe viajar más rápido que la velocidad más rápida soportada por la rejilla. Esta es la satisfacción física de las bien conocidas condiciones de Courant – Friedrichs – Lewy (CFL).

Por ejemplo, para una ecuación convectiva en una dimensión de la forma :

$$\partial_t f + U\,\partial_x f = 0 \ , \tag{1.59}$$

la condición CFL se lee de la siguiente manera :

$$C = \frac{U\,\Delta t}{\Delta x} < 1 \tag{1.60}$$

donde C es el número de Courant del esquema.

La interpretación física de esta desigualdad es que la velocidad física U no debe exceder la velocidad más alta soportada por la rejilla discreta $U_g \equiv \Delta x / \Delta t$.

Esto es tan solo un ejemplo para mostrar que LBE puede ser analizado bajo el criterio de las condiciones CFL.

Estabilidad Lineal : Ahora se considerara la ecuación BGK linealizada :

$$(\Delta_i + \omega)\,(f_i - f_i^0) = 0 \ , \tag{1.61}$$

donde f_i^0 es el equilibrio global uniforme y Δ_i es el operador de colisión discretizado. Puesto que $\Delta_i f_i^0 = 0$ entonces el termino puede ser sumado a la ecuación LBGK estándar (1.61).

Una transformada de Fourier de la ecuación (3.61) deriva en la siguiente relación de dispersión :

$$\prod_i (z_i - 1 + \omega) = 0 \ , \tag{1.62}$$

donde $z_i = e^{j(k_a c_{ia} - \omega)}$ y j denota la unidad imaginaria con el índice de velocidad discreta i.

La ecuación de arriba tiene un solo polo de multiplicidad b :

$$z_i = 1 - \omega \tag{1.63}$$

introduciendo la frecuencia ω dentro de la parte real Ω e imaginaria γ, la ecuación (1.63) se transforma en dos ecuaciones :

$$e^{-\gamma} \cos\theta_i = 1 - \Omega \ , \tag{1.64}$$

$$e^{-\gamma} sen\,\theta_i = 0 \tag{1.65}$$

donde $\theta_i = (k_a c_{ia} - \Omega)$ es la fase de los modos de Fourier propagándose a lo largo de la i-enésima dirección.

El resultado de la última ecuación es $\theta_i = 0$ (eso es , $\Omega = k_a c_{ia}$) el cual es precisamente la relación de dispersión de las ondas que viajan libres en el continuo.

Esto nos dice que el operador de colisión de la rejilla preserva la coherencia espacial puesto que no introduce ninguna distorsión del perfil espacial.

- **Eficiencia .**

En la actualidad los nuevos diseños de las arquitecturas computacionales hacen uso del concepto de 'reusabilidad de los datos' , el cual permite hacer un uso óptimo de los datos una vez que éstos han pasado de la memoria principal hacia los buffers internos del microprocesador [36]. Bajo este concepto, la información es procesada en el microprocesador con dos operaciones de punto flotante en vez de una sola como se hacia anteriormente. Esto reduce el tiempo total de procesamiento haciendo que la transferencia de datos de la memoria hacia el CPU ya no sea tan costosa.

Este concepto de reusabilidad de los datos permite que la sobrecarga de los modelos de LBE en las computadoras modernas no afecte en forma significativa el tiempo de ejecución. Esto es aún más significativo cuando se trata de computadoras paralelas.

Computadoras seriales o con eficiencia paralela combinadas con sistemas amigables son el conjunto básico para LBE que se usará en los próximos años.

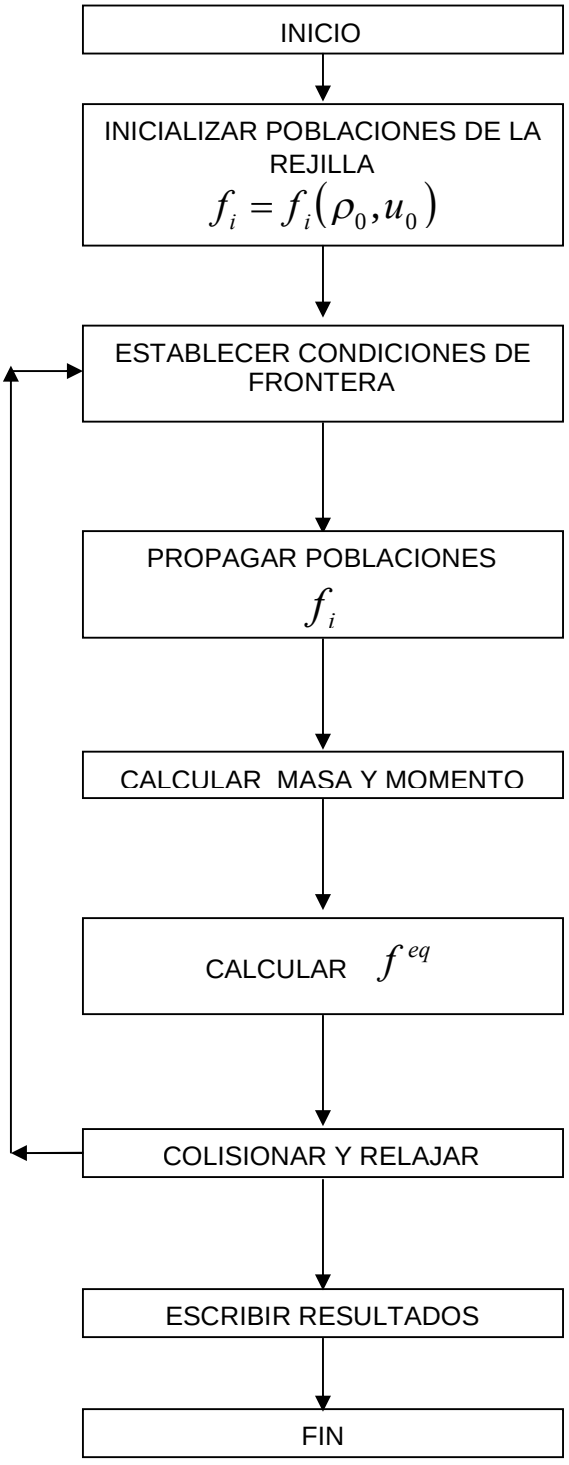
- **Flexibilidad .**

El punto principal es que tanto el espacio como el tiempo no son tratados en forma independiente, sino que ellos son unidos vía la condición : $dx_{i\,a} - c_{i\,a} dt = 0$.

Esto confiere a LBE la característica de aproximación adiabática. Bajo esta descripción, las moléculas son atadas a una rejilla en la cual se mueven paso a paso de un sitio a otro sin pararse en un lugar intermedio. En otras palabras ellas no pueden vivir en la medianía entre dos sitios, siempre deben estar en un sitio.

Esta característica hace que el esquema sea muy simple en términos de organización en estructura de datos y permite la discretización del operador de colisión sin usar alguna operación de punto flotante. Pero, esto también implica algunas limitantes como es la inhabilidad de aplicar LBE en casos más generales en donde se use una rejilla no uniforme.

1.9.- EJEMPLO DE SECUENCIA DE PASOS EN LA IMPLEMENTACIÓN COMPUTACIONAL DE UN MODELO DE REJILLA DE BOLTZMANN .



Inicializar las poblaciones de la rejilla con valores de densidad ρ_0 y velocidad u_0 iniciales.

Establecer condiciones de frontera bounce-back las paredes superior e inferior y periódicas para los Límites izquierdo y derecho de la rejilla.

Propagar las poblaciones f_i entre nodos contiguos en la rejilla.

$$f_i(x + e, t + \Delta t) = f(x, t)$$

Calcular valores de Masa y Momento .

$$\rho = \sum_i m\, f_i \qquad \rho\, u = \sum_i m\, e_i f_i$$

Calcular las funciones de Equilibrio f^{eq} :

$$f^{eq} = f^{eq}(\rho, u)$$

Colisionar y Relajar :

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

Nota: Ejemplo tomado de [27].

1.10.- SIMULACIÓN DEL COMPORTAMIENTO DE FLUIDOS NEWTONIANOS Y FLUIDOS NO NEWTONIANOS EN REJILLAS DE BOLTZMANN .

Esta de más mencionar la gran importancia que significa el tema de la viscosidad en el análisis del comportamiento de los fluidos. Los estudios de viscosidad son de gran utilidad en diversas áreas de la ciencia tales como la ingeniería mecánica (en el diseño de motores y en la selección de aceites lubricantes), en la medicina (para entender el flujo de la sangre en los ductos capilares) y en estudios básicos en la química, para determinación de masas molares de polímeros etc.

La viscosidad permite clasificar a los fluidos de acuerdo a su comportamiento viscoso como fluidos Newtonianos y fluidos No Newtonianos [28]. El estudio del comportamiento de los fluidos No Newtonianos es también de gran interés para diversas ramas de la ciencia y tecnología tales como la geofísica, la hidrología y las ciencias de los materiales entre otras. Aunque cabe destacar que en particular resulta útil para los fines de esta tesis, ya que el comportamiento de la sangre en forma integra (considerando todos sus componentes) es considerado como fluido No Newtoniano [29].

En un modelo de rejilla de Boltzmann la viscosidad cinemática esta relacionada en el operador de colisión Ω. En un modelo de rejilla hexagonal de LGCA, ésta se puede poner matemáticamente como :

$$n_i(\vec{x}+c_i,\ t+1) = n_i(\vec{x},\ t) + \Delta_i n_i\left(\vec{x},t\right) \tag{1.66}$$

donde $\Delta_i n_i\left(\vec{x},t\right)$ es el operador de colisión que describe como se espera que n_i cambie de acuerdo a las colisiones hechas en x.

En el modelo de rejilla de Boltzmann [Higuera y Jiménez, 1989] en vez de usar partículas discretas, se usan funciones de densidad $f_i(x,t)$ con un rango continuo entre 0 y 1. Estas funciones de densidad se propagan y colisionan con una ecuación de evolución en el tiempo similar a (1.66).

Si el sistema tiende al equilibrio, el operador de colisión $\Delta_i n_i\left(\vec{x},t\right)$ puede ser linealizado. Después de la linealización $\Delta_i n_i$ aparece como una matriz que actúa sobre las desviaciones de $f_i(x,t)$ de su equilibrio $f_i^{eq}(x,t)$, tal que la ecuación (1.66) toma la forma [30]:

$$f_i(x+c_i,\ t+1) = f_i(x,\ t) + \sum_{j=1}^{6} \Omega_{ij}\left(f_j(x,t) - f_j^{eq}(x,t)\right) \tag{1.67}$$

la matriz Ω_{ij} tiene solo dos eigenvalores para el caso partícular de un modelo D2Q9, los cuales no son predeterminados; uno es especificado como λ y determina la viscosidad cinemática v del fluido dada por la relación :

$$v = -\frac{1}{4}\left(\lambda + \frac{1}{2}\right) \tag{1.68}$$

donde λ es el parámetro de relajación y debe tomar valores en el rango de –2 a 0.

Los fluidos en los cuales su viscosidad v es independiente del movimiento del mismo ($\frac{\partial v_x}{\partial y}$), son considerados como fluidos Newtonianos (ver figura 1.24). Mientras que los fluidos cuya viscosidad depende del gradiente de velocidades ($v \neq cte$) son considerados como fluidos No Newtonianos.

Existen varios modelos de ecuaciones para describir el comportamiento de fluidos No Newtonianos [31]. Uno de los modelos más usados es el modelo de la ley de la potencia. En este modelo se dice que la viscosidad (llamada viscosidad aparente μ) depende del gradiente de velocidades de acuerdo con :

$$\mu = k(e)^n \tag{1.69}$$

donde ‘n’ es el índice de comportamiento del fluido, ‘k’ el índice de consistencia del fluido y ‘e’ denota el llamado segundo invariante de la velocidad del tensor de deformación, especificado como :

$$e = \sqrt{\left(\frac{\partial u_x}{\partial x}\right)^2 + \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)^2 / 2 + \left(\frac{\partial u_y}{\partial y}\right)^2} \tag{1.70}$$

Donde las componentes de la velocidad del fluido están dados por (u_x, u_y) .

Los valores de n=0 y n≠0 se asocian al comportamiento de fluidos Newtonianos y No Newtonianos respectivamente.

CAPÍTULO II

CREACIÓN DE UN MECANISMO DE FRONTERAS ABIERTAS EN REJILLAS DE BOLTZMANN

2.1.- ANÁLISIS DE OTROS MECANISMOS DE FRONTERAS ABIERTAS .

En el apartado 1.7 se vio explícitamente el funcionamiento de un mecanismo de fronteras abiertas propuesto por Succi [17]. Un segundo mecanismo propuesto por Qisu Zou y Xiaoyi He [32,33] consiste en manejar gradientes de presión o velocidad en la entrada y salida del ducto.

Para explicar el funcionamiento de este segundo mecanismo se parte del concepto de que un fluido en el interior de un ducto, en la práctica, puede ser movido por una diferencia de presiones entre la entrada y la salida del mismo. Bajo este concepto se necesita manejar una condición de frontera en la que se establece de antemano un valor de presión o velocidad en las fronteras del flujo. Estas fronteras no representan paredes sólidas o interfases entre dos fluidos, sino más bien son fronteras imaginarias dentro del dominio del flujo (en la entrada y salida del ducto) [34].

Puesto que en el método de rejillas de Boltzmann (LBM por sus siglas en ingles), la presión es relaciona con la densidad mediante la ecuación: $P = C_S^2 \rho$, donde C_S es la velocidad del sonido en el modelo, una especificación de diferencias en presiones se transforma en una especificación en diferencias de densidades en la entrada y salida del ducto [35,36].

- **Especificación de la presión del flujo en las fronteras .**

Tomando el esquema de nodo de la figura 2.1. Para un modelo de rejilla D2Q9 las ecuaciones (1.36) y (1.37) se pueden escribir como :

$$\text{(a)} \quad \sum_{i=0}^{8} f_i = \rho \ , \qquad \text{(b)} \quad \sum_{i=0}^{8} f_i c_i = \rho\, u \tag{2.1}$$

donde la densidad por nodo ρ y la velocidad macroscópica del flujo $u = (u_x, u_y)$ son definidas en términos de la función de distribución de la partícula.

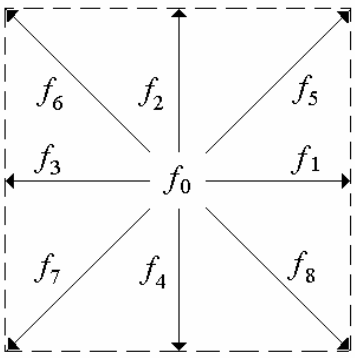


Figura 2.1. Esquema de nodo para un modelo de rejilla D2Q9.

Partiendo de la hipótesis de que un valor de densidad de entrada ρ_{in} es dado y que una velocidad de flujo $u_y = 0$ es especificada como condición de entrada. Después del proceso de propagación, las poblaciones: f_2, f_3, f_4, f_6, f_7 son conocidas. (ver figura 2.2)

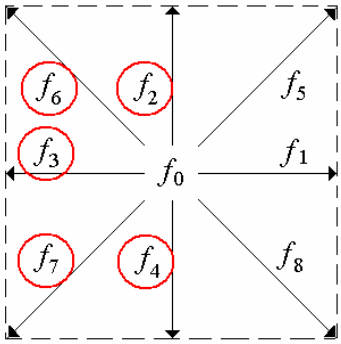


Figura 2.2. Poblaciones conocidas después del proceso de propagación.

Por lo que se necesitan determinar el valor de u_x y los valores de las poblaciones : f_1, f_5 y f_8 . Usando las ecuaciones (2.1) se obtiene lo siguiente:

$$f_1 + f_5 + f_8 = \rho_{in} - \left(f_0 + f_2 + f_3 + f_4 + f_6 + f_7\right), \tag{2.2}$$

$$f_1 + f_5 + f_8 = \rho_{in}\, u_x + \left(f_3 + f_6 + f_7\right), \tag{2.3}$$

$$f_8 - f_5 = f_2 - f_4 + f_6 - f_7 \tag{2.4}$$

Consistentemente con las ecuaciones (2.2) y (2.3) tenemos:

$$u_x = 1 - \frac{\left[f_0 + f_2 + f_4 + 2\left(f_3 + f_6 + f_7\right)\right]}{\rho_{in}} \tag{2.5}$$

Asumiendo que bounce-back es una regla valida para la parte de no-equilibrio de la distribución normal de la partícula en la entrada (para este caso $f_1 - f_1^{(eq)} = f_3 - f_3^{(eq)}$). Con f_1 conocida, f_5 y f_8 pueden ser determinadas por :

$$f_1 = f_3 + \frac{2}{3}\rho_{in}u_x \ ,$$
$$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x \ , \qquad (2.6)$$
$$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x$$

Los nodos de las esquinas de la entrada reciben un trato especial . Tomando el nodo de la esquina de abajo de la entrada como ejemplo, después del proceso de propagación f_3, f_4, f_7 son conocidas y el valor de densidad 'ρ' es especificado (ver figura 2.3).

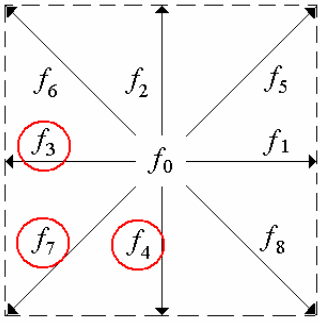


Figura 2.3. Nodo de la esquina inferior de la entrada y las poblaciones conocidas después del proceso de propagación.

Se necesitan determinar entonces f_1, f_2, f_5, f_6 y f_8 . Usándo la regla bounce-back para la parte de no-equilibrio de la distribución normal de la partícula para la entrada se obtiene :

$$f_1 = f_3 + \left(f_1^{(eq)} - f_3^{(eq)}\right) = f_3 \ , \qquad (2.7)$$

$$f_2 = f_4 + \left(f_1^{(eq)} - f_3^{(eq)}\right) = f_4 \qquad (2.8)$$

Usando el cálculo de f_1 y f_2 en la ecuación (2.1) , se tiene que :

$$f_5 = f_7 \ , \qquad (2.9)$$

$$f_6 = f_8 = \frac{1}{2}\left[\rho_{in} - (f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_7)\right] \qquad (2.10)$$

Un procedimiento similar puede ser aplicado en el nodo de la esquina superior de la entrada y en los nodos de la salida, con los cuales se toma el valor de densidad deseado de salida, incluyendo los nodos de las equinas superior e inferior de la salida.

- **Especificación de la velocidad del flujo en las fronteras .**

El valor de las velocidades u_x, u_y son especificados para la entrada y la salida del ducto. Tomando como ejemplo un nodo de la entrada del ducto, después del proceso de propagación el valor de las poblaciones f_2, f_4, f_3, f_6, f_7 son conocidos (ver figura 2.2).

Entonces se necesita determinar el valor de densidad ρ y el valor de las poblaciones f_1, f_5 y f_8 .

Usando las ecuaciones (2.1) se obtiene :

$$f_2 + f_5 + f_6 = \rho - \left(f_0 + f_1 + f_3 + f_4 + f_7 + f_8\right), \tag{2.11}$$

$$f_5 - f_6 = \rho\, u_x - \left(f_1 - f_3 - f_7 + f_8\right), \tag{2.12}$$

$$f_2 + f_5 + f_6 = \rho\, u_y + \left(f_4 + f_7 + f_8\right) \tag{2.13}$$

En consistencia con las ecuaciones (2.11) y (2.12) se tiene:

$$\rho = \frac{1}{1-u_x}\left[f_0 + f_2 + f_4 + 2\left(f_3 + f_6 + f_7\right)\right] \tag{2.14}$$

Nótese como las ecuaciones (2.11), (2.12) y (2.13) están en función de los valores de velociadad deseados de entrada u_x y u_y, y del valor de densidad del nodo deteminado mediante la ecuación (2.14). A diferencia de las ecuaciones (2.2), (2.3) y (2.4), las cuales inversamente estan en función de un valor de densidad deseado de entrada (ρ_{in}) y de un valor de velocidad u_x que es calculado por medio de la ecuación (2.5).

Asumiendo que bounce-back es una regla valida para la parte de no-equilibrio de la distribución normal de la partícula en la entrada (para este caso $f_1 - f_1^{(eq)} = f_3 - f_3^{(eq)}$). Con f_1 conocida, f_5 y f_8 pueden ser determinadas por :

$$f_1 = f_3 + \frac{2}{3}\rho\, u_x \ ,$$
$$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{2}\rho\, u_y + \frac{1}{6}\rho\, u_x \ , \qquad (2.15)$$
$$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{2}\rho\, u_y + \frac{1}{6}\rho\, u_x$$

El efecto de especificar la velocidad en la entrada es similar a la especificación de la presión en la entrada, ya que ambas se pueden manejar como una diferencia de densidades en los nodos de la entrada y de la salida del flujo.

Los nodos de las esquinas de la entrada reciben un trato especial . Tomando el nodo de la esquina de abajo de la entrada como ejemplo, después del proceso de propagación f_1, f_2, f_5, f_6 y f_8 necesitan ser determinados (ver figura 2.3).

Usando bounce-back en las distribuciones normales se obtiene :

$$f_1 = f_3 \quad , \quad f_2 = f_4 \qquad (2.16)$$

y mediante las expresiones de momento en x,y se obtiene :

$$f_5 - f_6 + f_8 = -(f_1 - f_3 - f_7) = f_7 \qquad (2.17)$$

$$f_5 + f_6 - f_8 = -(f_2 - f_4 - f_7) = f_7 \qquad (2.18)$$

con las cuales se obtienen :

$$f_5 = f_7 \qquad (2.19)$$

$$f_6 = f_8 = \frac{1}{2}\left[\rho - (f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_7)\right] \qquad (2.20)$$

El valor de densidad ρ puede ser tomado del valor de densidad ρ de los nodos vecinos en donde los valores de velocidad son especificados.

La situación es similar para el resto de los nodos de las esquinas. Y en general la salida es tratada en forma similar a la entrada.

- **Limitaciones del modelo de gradientes de velocidad y/o presión .**

Como se explicó para este modelo en el apartado anterior y tomando como ejemplo a un nodo que se encuentre en la medianía de la entrada del ducto (llamémoslo nodo x) , después del proceso de propagación los valores de las poblaciones f_2, f_4, f_6, f_7, f_3 son conocidos y los valores de f_5, f_1 y f_8 (ver figura 2.2) se deben de determinar de acuerdo con las ecuaciones (2.6) o (2.15), dependiendo del caso.

De acuerdo con las ecuaciones (2.1) se desprende que los valores de f_5, f_1 y f_8 deben de asegurar la conservación de masa y momento a nivel macroscópico.

Como los valores de las poblaciones f_5, f_1 y f_8 son establecidos con base a parámetros libres de velocidad y/o presión de entrada y salida que van de acuerdo con el estado físico de un sistema real, las poblaciones f_5, f_1 y f_8 pueden tomar cualquier valor, lo que puede conducir a las siguientes situaciones :

Análisis (1) :

- Que el nodo x experimente de acuerdo con (2.1) una pérdida de densidad local respecto al valor que tenía antes del proceso de propagación (P).
- Que el nodo x experimente de acuerdo con (2.1) una ganancia de densidad local respecto al valor que tenía antes del proceso de propagación (G).
- Que el nodo x experimente de acuerdo con (2.1) una conservación del mismo valor de densidad local respecto al valor que tenía antes del proceso de propagación (C).

Como el mismo trato es aplicado a los nodos que conforman la salida del ducto, exceptuando los nodos de las equinas, entonces para un modelo de rejilla D2Q9 se tiene lo siguiente :

$$\rho_{ent} = \sum_{y=1}^{Ly-2} \sum_{i=0}^{8} f_i(1, y) \ , \qquad \rho_{sal} = \sum_{y=1}^{Ly-2} \sum_{i=0}^{8} f_i(Lx, y) \tag{2.21}$$

entendiéndose a ρ_{ent} y a ρ_{sal} como la acumulación de densidades de los nodos que conforman la entrada y salida del ducto respectivamente. Lx y Ly son los límites físicos de la rejilla.

De acuerdo con el análisis (1), un nodo que conforme la entrada o de la salida del ducto puede experimentar variaciones (ganancias o perdidas) de densidad local, por lo que ρ_{ent} y ρ_{sal} pueden experimentar consecuentemente variaciones con respecto al valor que tenían antes de la propagación. Esto lleva a tener las situaciones expresadas en la tabla 2.1.

ρ_{ent}	ρ_{sal}
P	P
P	G
P	C
G	P
G	G
G	C
C	P
C	G
C	C

Tabla 2.1. Posibles estados del valor de densidad de entrada y de salida del ducto. P ≡ Perdida de densidad ; G ≡ Ganancia de densidad ; C ≡ Conservación del mismo valor de densidad respecto al valor que se tenía antes del proceso de propagación.

Donde solo la configuración [C][C] asegura que exista una conservación de masa y momento a nivel macroscópico. Las configuraciones [P][G] y [G][P] también lo pueden hacer siempre y cuando cumplan la condición G - P = 0. Las demás configuraciones siempre llevaran a estados del sistema donde se tienen variaciones de masa y momento a nivel macroscópico.

Dado que solo existen tres configuraciones que aseguran una estabilidad del sistema a nivel macroscópico (esto es, existe una conservación de masa y momento) y que el rango de valores de presión y velocidades que pueden generar estos tres posibles estados es limitado con respecto a cualquier valor arbitrario, se desprende la siguiente conclusión: El modelo de gradientes de presión y/o velocidades es estable a nivel macroscópico (existe una conservación de masa y momento) solo para un rango específico de valores de presión y velocidades de entrada y de salida, lo que impide que el modelo sea viable para simular toda situación del mundo real.

Un comentario similar es especificado como autocrítica por Qisu Zou y Xiaoyi He (autóres del tema) en su artículo " On Pressure and Velocity Flow Boundary Conditions for the Lattice Boltzmann BGK Model " [32]. Los cuales declaran que: "en la entrada, la ecuación (2.1-a) quizá no sea igual al valor de densidad especificado después del proceso de propagación. Esto es una inconsistencia que causa errores en las simulaciones, lo cual puede ser un área de oportunidad de mejora para el método".

- **Limitaciones del modelo de Succi .**

De acuerdo con Succi la ecuación (1.54) permite asegurar una conservación de masa y momento a nivel macroscópico mediante la reflexión de las partículas hacia atrás con una probabilidad 'r'. Succi también indica que cuando el flujo de salida es igual a la velocidad de entrada ($u = u_{in}$) el resultado de la evaluación de esta ecuación es uno (r=1), que indica que la reflexión no es necesaria debido a que el flujo es automáticamente estable (no presenta excesos de velocidad). Mientras que cuando el flujo de salida es mayor que la velocidad de entrada ($u > u_{in}$) el resultado es menor a uno (r<1) que indica que se debe hacer reflexión para absorber el exceso de velocidad que es producido. (ver punto 1.7)

Bajo este esquema podemos ver a la ecuación (1.54) como un sistema de balance de masa que la mantiene constante a nivel macroscópico. También es claro ver que la ecuación está solo en función de los valores de la velocidad del flujo de entrada y de los valores de la velocidad del flujo de salida y no toma en cuenta la geometría interna del ducto. Lo que la hace ser insensible a los cambios de diámetro interno de los ductos. y siendo consistente con el principio de Bernoulli que establece que un ducto que presenta variaciones en su diámetro interno también presenta variaciones de velocidad en el flujo.

Por lo que su valides solo se limita a los casos en los que se trabaja con ductos de geometría interna recta (paredes lisas) .

Tómese el siguiente caso como ejemplo para ilustrar esta deficiencia.

Un ducto con geometría interna recta y un ducto con geometría interna cónica (ver figura 2.4) obtienen el mismo resultado en la evaluación de la ecuación (1.54) (r<1) bajo la base de $u > u_{in}$ (velocidad de salida mayor que la velocidad de entrada). La diferencia esta en que en el primer caso cuando el ducto es de geometría interna recta se debe hacer reflexión para absorber el exceso de velocidad $u - u_{in}$. Mientras que para el segundo caso cuando el ducto es de geometría interna cónica, no debe haber reflexión debido a que el exceso de velocidad es provocado por la diferencia de diámetros internos en el ducto.

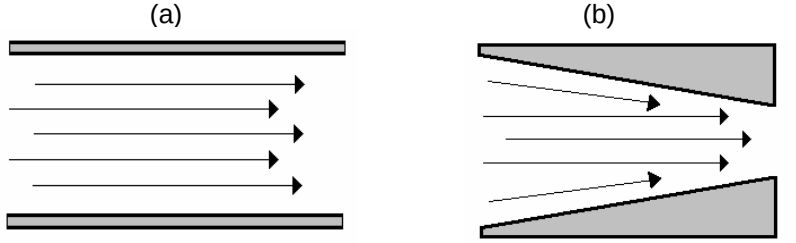


Figura 2.4. (a) ducto con geometría interna recta (b) ducto con geometría interna cónica .

Este inconveniente impide que el modelo de Succi proporcione resultados confiables cuando se analizan ductos con geometría interna no recta, como es el caso de una arteriola. Es por esta razón que el modelo de Succi resultó ser deficiente para los objetivos que se persiguen en esta tesis.

2.2.- NUEVO MECANISMO DE FRONTERAS ABIERTAS PROPUESTO .

Dados los inconvenientes encontrados en los mecanismo de Succi [17] y de Qisu Zou y Xiaoyi He [32] para los objetivos de esta tesis, se propone manejar un nuevo mecanismo de fronteras abiertas para un modelo de rejilla D2Q9 que superará éstas limitantes. Por un lado éste nuevo mecanismo es estable, ésto es, presenta una conservación de masa y momento a nivel macroscópico. Por otro lado es sensible a la geometría interna del ducto por el cual fluye el fluido. Características que mejoran al modelo y le confieren mayor capacidad para simular situciones del mundo real.

Tomando como base el ejemplo de secuencia de pasos en la implementación computacional de un modelo de rejilla de Boltzmann descrito en el apartado 1.9. La secuencia de pasos muestra 7 procesos básicos.

Primer Proceso: El primero proceso es inicializar las poblaciones de la rejilla con valores de densidad ρ_0 y velocidad e_0 iniciales. Para un modelo de rejilla D2Q9 la distribución de las velocidades iniciales es 4/9 para e_0 , 1/9 para e_i con valores de i=1,2,3,4 y 1/36 para e_i con valores de i=5,6,7,8, de acuerdo con los valores establecidos como parámetro en la tabla 3.1. Ver figura 2.5.

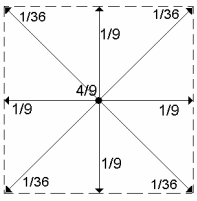


Figura 2.5. Valores de velocidad iniciales para las poblaciones de un modelo de rejilla D2Q9.

Los valores de densidad ρ_0 son distribuidos proporcionalmente de acuerdo con los valores de velocidad inicial e_0 asignados. Ver figura 2.6.

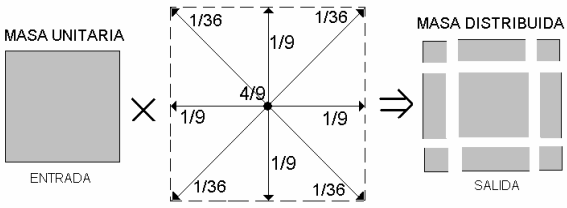


Figura 2.6. Distribución de densidad proporcional a los valores de velocidad iniciales para las poblaciones de un modelo de rejilla D2Q9.

El siguiente segmento de código hecho en Delphi versión 7.0 ilustra este proceso y se encuentra encapsulado en un procedimiento denominado 'inicializa rejilla' , el cual recibe como parámetro el valor de masa unitario con el que se inicializa cada nodo de la rejilla.

```
Procedure iniciliza_rejilla ( MasaUnitaria:real ) ;
Var     t_0,t_1,t_2:real;
        x:integer;
Begin
        t_0:= MasaUnitaria * 4/9;  t_1:= MasaUnitaria / 9;  t_2:= MasaUnitaria /36;
        for x:=1 to lx do
                for y:=1 to ly do begin
                        node[0,x,y]:=t_0;   node[1,x,y]:=t_1;   node[2,x,y]:=t_1;
                        node[3,x,y]:=t_1;   node[4,x,y]:=t_1;   node[5,x,y]:=t_2;
                        node[6,x,y]:=t_2;   node[7,x,y]:=t_2;   node[8,x,y]:=t_2;
                end;
End;
```

Segundo Proceso: El segundo proceso es establecer condiciones de frontera. Para esto los nodos que conforman la limites superior e inferior de la rejilla reciben un trato especial que se conoce como condición de frontera. El tipo de condición de frontera aplicado en este caso es 'No deslizante' y tiene la finalidad de simular el comportamiento que manifiestan las partículas del fluido real al colisionar con las paredes del ducto por el cual fluye (ver apartado 1.7).

Como ya se comento en el apartado 1.7, con esta condición se logra, mediante la reflexión de los valores de las poblaciones f_i, que las componentes normal y tangencial de las velocidades e_i de las partículas del fluido que están en contacto directo con las paredes del ducto, se anulen (ver figura 1.18).

El siguiente segmento de código en Delphi 7.0 ilustra como el proceso 'Bounce-Back' es manejado computacionalmente .

```
Procedure BounceBack;
Var     x,y,k:integer;
Begin
        for x:=1 to lx do
            for k:=0 to 8 do begin
                n_hlp[k,x,1]:= node[k,x,1];
                n_hlp[k,x,ly]:=node[k,x,ly];
            end;
```

```
        for x:=1 to lx do begin
                node[2,x,1]:=n_hlp[4,x,1];
                node[4,x,1]:=n_hlp[2,x,1];
                node[5,x,1]:=n_hlp[7,x,1];
                node[6,x,1]:=n_hlp[8,x,1];
                node[7,x,1]:=n_hlp[5,x,1];
                node[8,x,1]:=n_hlp[6,x,1];

                node[2,x,ly]:=n_hlp[4,x,ly];
                node[4,x,ly]:=n_hlp[2,x,ly];
                node[5,x,ly]:=n_hlp[7,x,ly];
                node[6,x,ly]:=n_hlp[8,x,ly];
                node[7,x,ly]:=n_hlp[5,x,ly];
                node[8,x,ly]:=n_hlp[6,x,ly];
        end;
End;
```

Donde n_hlp es una estructura auxiliar que es utilizada para almacenar temporalmente los valores de la estructura node y no perderlos durante el proceso de reflexión.

Tercer Proceso: El tercer proceso es propagar el valor de las poblaciones f_i. Para un modelo de rejilla D2Q9 de Lx nodos de largo y Ly nodos de ancho, la propagación de las poblaciones esta dada por :

$$
\begin{aligned}
f_i(x,y) &= f_i(x,y) \quad , \; i=0 \\
f_i(xE,y) &= f_i(x,y) \quad , \; i=1 \\
f_i(x,yN) &= f_i(x,y) \quad , \; i=2 \\
f_i(\mathrm{xW},\mathrm{y}) &= f_i(x,y) \quad , \; i=3 \\
f_i(x,yS) &= f_i(x,y) \quad , \; i=4 \\
f_i(xE,yN) &= f_i(x,y) \quad , \; i=5 \\
f_i(xW,yN) &= f_i(x,y) \quad , \; i=6 \\
f_i(xW,yS) &= f_i(x,y) \quad , \; i=7 \\
f_i(xE,yS) &= f_i(x,y) \quad , \; i=8
\end{aligned}
$$

con valores definidos de $yN = (y \bmod \mathrm{Ly}) + 1$, $x\mathrm{E} = (x \bmod \mathrm{Lx}) + 1$ $yS = \mathrm{Ly} - [(\mathrm{Ly} + 1 - y) \bmod \mathrm{Ly}]$ y $x\mathrm{W} = \mathrm{Lx} - [(\mathrm{Lx} + 1 - x) \bmod \mathrm{Lx}]$.

La figura 2.7 ilustra la propagación de los valores para un determinado nodo (x,y) de la rejilla.

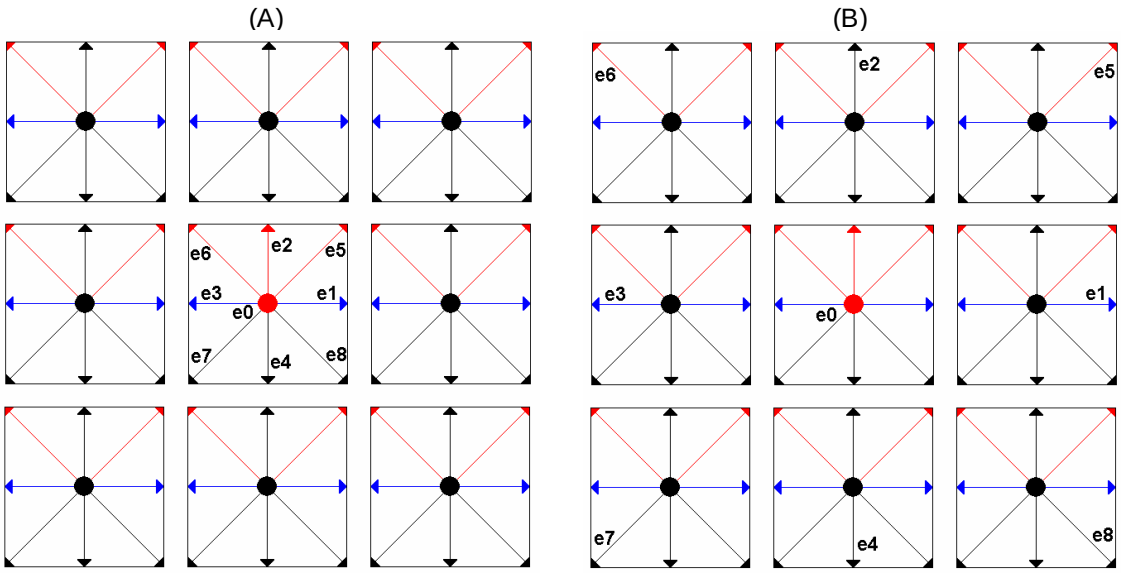


Figura 2.7. La figura (A) y (B) muestra los valores de las poblaciones f_i antes y después de la propagación respectivamente.

El siguiente segmento de código en Delphi 7.0 ilustra el proceso de propagación .

```
Procedure Propagacion;
Var    x,y,y_n,x_e,y_s,x_w:integer;
Begin
        for x:=1 to lx do
                for y:=1 to ly do begin
                        y_n:=(y mod ly) +1;
                        x_e:=(x mod lx) +1;
                        y_s:=ly-((ly+1-y) mod ly);
                        x_w:=lx-((lx+1-x) mod lx);
                        n_hlp[0,x,y]:=node[0,x,y];
                        n_hlp[1,x_e,y]:=node[1,x,y];
                        n_hlp[2,x,y_n]:=node[2,x,y];
                        n_hlp[3,x_w,y]:=node[3,x,y];
                        n_hlp[4,x,y_s]:=node[4,x,y];
                        n_hlp[5,x_e,y_n]:=node[5,x,y];
                        n_hlp[6,x_w,y_n]:=node[6,x,y];
                        n_hlp[7,x_w,y_s]:=node[7,x,y];
                        n_hlp[8,x_e,y_s]:=node[8,x,y];
                end;
end;
```

Cuarto Proceso: El cuarto proceso es calcular el valor de la masa en el sistema. Este es un proceso simple, el cual consiste en sumar el valor de las poblaciones de cada nodo en la rejilla para calcular primero el valor de densidad local y después sumar el valor de las densidades locales de todos los nodos que integran la rejilla para obtener el valor de masa en el sistema. El siguiente segmento de código en Delphi 7.0 ilustra este proceso para un modelo de rejilla D2Q9.

```
Procedure Calcula_Masa;
Var     x,y,k:integer;
        masa, densidad_local:real;
Begin
        masa:=0;
        for x:=1 to lx do
                for y:=1 to ly do begin
                    densidad_local:=0
                    for k:=0 to 8 do densidad_local:= densidad_local +node[k,x,y];
                    masa:=masa+ densidad_local;
                end;
        showmessage(floattostr(masa));
end;
```

Para mantener la conservación de masa en el sistema se propone añadir un proceso adicional el cual consiste en manejar una columna de nodos auxiliares extra (buffers) en el lado derecho de la rejilla, considerando que el flujo fluye de izquierda a derecha en el sistema (ver figura 2.8).

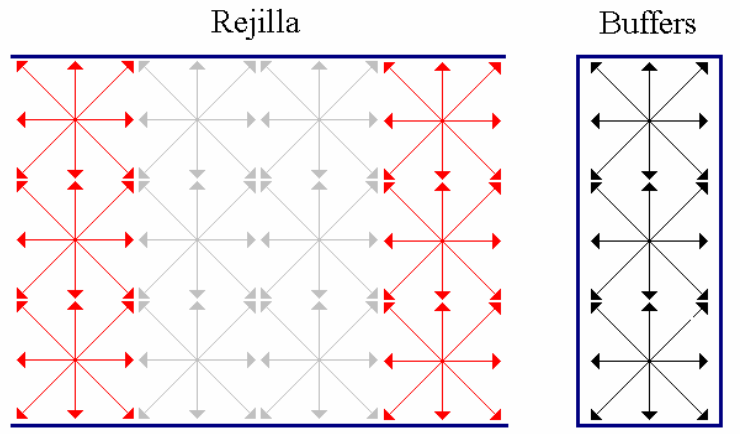


Figura 2.8. Columna de nodos extra (buffers) añadidos del lado derecho de la rejilla.

La idea es calcular la diferencia de masas (gradiente de masa) que existe en la última columna de nodos de la rejilla antes del proceso de propagación y después del proceso de propagación y almacenar esta gradiente en la columna de nodos extra (buffers).

Los siguientes segmentos de código en Delphi 7.0 ilutran este proceso de cópia de densidad antes y después del proceso de propagación.

```
Procedure Copia_Densidad_Antes;
Var     y,k:integer;
Begin
        for y:=1 to ly do
            for k:=0 to 8 do densidad_antes[k,y]:= node[k,Lx,y];
end;

Procedure Copia_Densidad_Despues;
Var     y,k:integer;
Begin
        for y:=1 to ly do
            for k:=0 to 8 do densidad_despues[k,y]:= node[k,Lx,y];
end;
```

Donde ‘densidad_antes’ y ‘densidad_despues’ son variables globales en el sistema.

Partiendo de la base de que el sistema presenta conservación de masa antes de la propagación. Si existe un gradiente de masa en la salida después de la propagación, estaría indicando que hubo movimiento de flujo y que éste lo produjo, por lo tanto a este exceso de masa en la salida le debe corresponder un déficit de masa de igual proporción en la entrada (primer columna de la rejilla), por lo que para compensar esta perdida debe haber una retroalimentación del exceso de masa en la salida (buffers) en la entrada o dicho de otra forma debe haber un balance de masa en el sistema de fronteras abiertas. (ver figura 2.9)

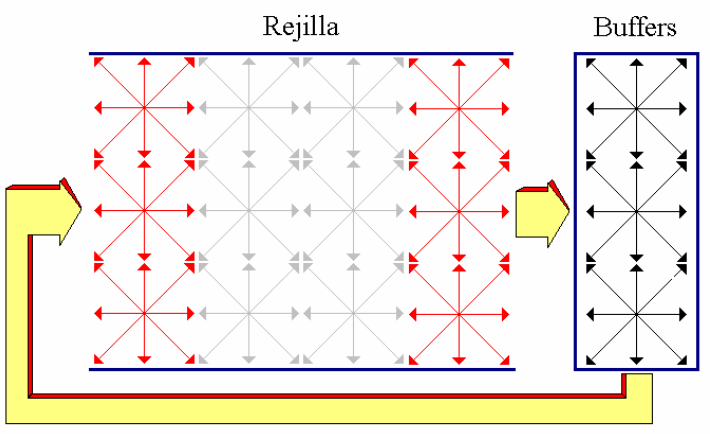


Figura 2.9. Retroalimentación de la diferencia de masas en la salida hacia la entrada.

Con este mecanismo de balance de masa se asegura una conservación de masa en el sistema a nivel macroscópico.

El siguiente segmento de código en Delphi 7.0 muestra como es llevado a cabo computacionalmente este proceso de balance de masa con base a los valores de 'densidad_antes' y 'densidad_despues' calculados previamente.

```
Procedure BalancedeMasa;
Var   y,k:integer;
Begin
      for y:=1 to ly do
            for k:=0 to 8 do
                DifVect[k,y]:=densidad_antes[k,y]-densidad_despues[k,y];

      for y:=1 to ly do begin
            node[0,1,y]:= node [0,1,y]+DifVect[0,y];
            node [1,1,y]:= node [1,1,y]+DifVect[3,y];
            node [2,1,y]:= node [2,1,y]+DifVect[2,y];
            node [3,1,y]:= node [3,1,y]+DifVect[1,y];
            node [4,1,y]:= node [4,1,y]+DifVect[4,y];
            node [5,1,y]:= node [5,1,y]+DifVect[7,y];
            node [6,1,y]:= node [6,1,y]+DifVect[8,y];
            node [7,1,y]:= node [7,1,y]+DifVect[5,y];
            node [8,1,y]:= node [8,1,y]+DifVect[6,y];
      end;
End;
```

Quinto Proceso: El quinto proceso consiste en calcular las funciones de equilibrio f^{eq}. Para cubrir las deficiencias de sensibilidad de geometría interna del mecanismo de Succi se propone manejar una matriz auxiliar de gradientes de velocidad de dimensiones iguales a las de la rejilla. Para esto se parte de los siguientes preceptos :

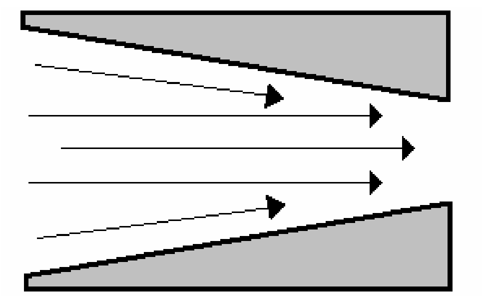
(1) Se deben de conocer de antemano los valores a establecer de las velocidades de entrada y salida del ducto.

(2) Se debe de conocer de antemano la geometría interna del ducto.

Dados estos preceptos la matriz de gradientes de velocidad debe contener el espectro de valores completo de gradientes de velocidad que existe entre la velocidad establecida de entrada y la velocidad establecida de salida.

La figura 2.10 ilustra una matriz muestra con el espectro completo de gradientes de velocidad en x entre una velocidad de entrada deseada ejemplo (30) y una velocidad de salida deseada ejemplo (08). La matriz debe ser adaptada a la geometría interna del ducto .

(a)



(b)

30																						
30	29	28	27	26																		
30	29	28	27	26	25	24	23	22														
30	29	28	27	26	25	24	23	22	21	20	19	18										
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14						
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11			
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11			
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14						
30	29	28	27	26	25	24	23	22	21	20	19	18										
30	29	28	27	26	25	24	23	22														
30	29	28	27	26																		
30																						

Figura 2.10. (a) ducto con forma cónica. (b) matriz de gradientes de velocidad correspondiente.

La finalidad de tener una matriz de gradientes de velocidad es conocer la distribución exacta de velocidades que en principio y en cada punto (x,y) debe existir en la rejilla.

Definiendo a la matriz de gradientes de velocidad como Δ , se tiene que las anotaciones $\Delta_X(x,y)$ y $\Delta_Y(x,y)$ corresponden a las componentes en ‘x’ y en ‘y’ de las velocidades, donde cada componente es también tratada en forma matricial. Así para el ejemplo de la figura 2.10 en el que solo se muestra la componente en ‘x’ de las velocidades de entrada y salida deseadas, su representación es $\Delta_X(x,y)$. La forma de obtener $\Delta_Y(x,y)$ es similar a como se obtiene $\Delta_X(x,y)$.

El siguiente segmento de código en Delphi 7.0 muestra como es calculada la matriz de gradientes de velocidad (MatrizDeltas).

```
Procedure CalculoMatrizDeltas ( MasaLocal : real ) ;
Var   DeltaX56, DeltaX85, DeltaX13, DeltaX42, DeltaX87, DeltaX76 :real;
      T1, T2 : real;
       x, y : Integer;
Begin
        DeltaX56 := (NodoVelUx[5] - NodoVelUx[6]) / Lx;
        DeltaY85 := (NodoVelUy[8] - NodoVelUy[5]) / Ly;
        DeltaX13 := (NodoVelUx[1] -  NodoVelUx[3]) / Lx;
        DeltaY42 := (NodoVelUy[4] - NodoVelUy[2] ) / Ly;
        DeltaX87 := (NodoVelUx[8] -  NodoVelUx[7]) / Lx;
        DeltaY76 := (NodoVelUy[7] - NodoVelUy[6] ) / Ly;
```

```
        VelUx1:= NodoVelUx[6];
        VelUx2:= NodoVelUx[3];
        VelUx3:= NodoVelUx[7];
        T1:= MasaLocal *1/9;
        T2:= MasaLocal *1/36;
        for x:=1 to  lx do begin
                for y:=1 to ly do begin
                        MatrizDeltas[x,y]. X56:= VelUx1 * T2;
                        MatrizDeltas[x,y]. X13:= VelUx2 * T1;
                        MatrizDeltas[x,y]. X87:= VelUx3 * T2;
                end;
                VelUx1:=VelUx1  - DeltaX56;
                VelUx2:=VelUx2  - DeltaX13;
                VelUx3:=VelUx3  - DeltaX87;
        end;

        VelUy1:= NodoVelUy[5];
        VelUy2:= NodoVelUy[2];
        VelUy3:= NodoVelUy[6];
        T1:= MasaLocal *1/9;
        T2:= MasaLocal *1/36;
        for y:=1 to  ly do begin
                for x:=1 to lx do begin
                        MatrizDeltas[x,y]. Y58:= VelUy1 * T2;
                        MatrizDeltas[x,y]. Y24:= VelUy2 * T1;
                        MatrizDeltas[x,y]. Y67:= VelUy3 * T2;
                end;
                VelUy1:=VelUy1  - DeltaY85;
                VelUy2:=VelUy2  - DeltaY42;
                VelUy3:=VelUy3  - DeltaY76;
        end;
End;
```

Donde las componentes de las velocidades deseadas de entrada y salida son almacenadas previamente en estructuras auxiliares : NodoVelUx y NodoVelUy.

Partiendo de esta base, se tiene que para un modelo de rejilla D2Q9, como el que se muestra en la figura 2.11, el cálculo del valor de las poblaciones f_i es el siguiente :

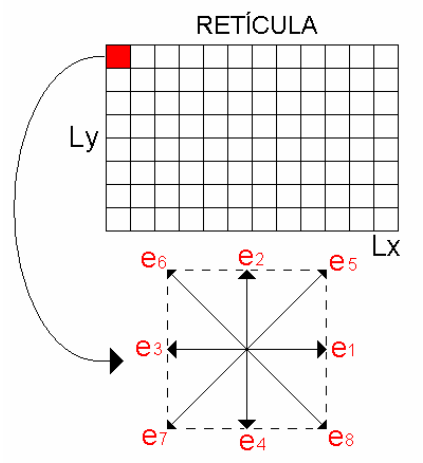


Figura 2.11. Modelo rejilla de Boltzmann D2Q9.

Para un modelo BGK el cálculo de las poblaciones f_i es definido por la ecuación [45] :

$$f_i\left(x+e_i\,,t+1\right)-f_i\left(x,t\right)=-\frac{1}{\tau}\left[f_i\left(x\,,t\right)-f_i^{(eq)}\left(x\,,t\right)\right]\,,\quad i=0,\ldots,8, \tag{2.22}$$

de donde se tiene que para un determinado nodo (x,y) de la rejilla (ver figura 2.11) el equilibrio local es dado por [45] :

$$f_0^{(eq)}=\frac{4}{9}\rho\left[1-\frac{3}{2}\mathrm{u}\cdot\mathrm{u}\right],$$

$$f_i^{(eq)}=\frac{1}{9}\rho\left[1+3(\mathrm{e}_i\cdot\mathrm{u})+\frac{9}{2}(\mathrm{e}_i\cdot\mathrm{u})^2-\frac{3}{2}\mathrm{u}\cdot\mathrm{u}\right],\qquad i=1,2,3,4 \tag{2.23}$$

$$f_i^{(eq)}=\frac{1}{36}\rho\left[1+3(\mathrm{e}_i\cdot\mathrm{u})+\frac{9}{2}(\mathrm{e}_i\cdot\mathrm{u})^2-\frac{3}{2}\mathrm{u}\cdot\mathrm{u}\right],\qquad i=5,6,7,8$$

con una velocidad neta de flujo ‘u’ definida como :

$$\mathrm{U}^2=\mathrm{U}_\mathrm{x}^2+\mathrm{U}_\mathrm{y}^2 \tag{2.24}$$

donde las componentes de la velocidad neta $\mathrm{U_x}$ y $\mathrm{U_y}$ son definidas como :

$$\mathrm{Ux}=\left(e_1(x,y)+e_5(x,y)+e_8(x,y)\right)-\left(e_3(x,y)+e_6(x,y)+e_7(x,y)\right)+\Delta_x(x,y) \tag{2.25}$$

$$\mathrm{Uy}=\left(e_2(x,y)+e_5(x,y)+e_6(x,y)\right)-\left(e_4(x,y)+e_7(x,y)+e_8(x,y)\right)+\Delta_y(x,y) \tag{2.26}$$

con : $1\le x\le \mathrm{Lx}$ y $1\le y\le \mathrm{Ly}$.

Obsérvese como los componentes $\Delta_X(x,y)$ y $\Delta_Y(x,y)$ de la matriz de gradientes de velocidad afectan el cálculo de la velocidad neta de flujo que se encuentra en cada punto (x,y) de la rejilla. Lo que provoca que la velocidad neta de flujo se integre de dos componentes, por un lado la componente producida por el propio sistema, la cual esta en función del comportamiento de los nodos vecinos (sensibilidad a la geometría del ducto en términos macroscópicos) y por otro lado la componente de la matriz de gradientes que le indica al sistema cual es el valor de velocidad deseado para ese punto en particular (x,y) y que se fijo de antemano dadas las condiciones deseadas de velocidad de entrada y salida en el ducto.

Análisis Particular.

Como se puede apreciar, alterar la velocidad neta del flujo puede llevar a dos situaciones:

1). *Afectar el valor de densidad local.*

En este punto es lógico pensar que si se altaran las funciones de distribución de equilibrio f^{eq} en un $\pm \Delta f^{eq}$, el valor de las poblaciones f_i resulta afectado en un $\pm \Delta f_i$, lo que conlleva a pensar que el valor de masa local $\sum_{i=0}^{8} f_i$ resulta alterado respecto al valor que tenía antes del proceso de relajación. En principio ésto es cierto. Sin embargo, aunque existen pepeños cambios en el valor de densidad local, que pueden ser aumentos o decrementos en cada nodo de la rejilla, el valor de masa global en la reajilla permanece constante después del proceso de relajación, según lo observado en la práctica. Esto debido a que las perdidas de masa que presentan algunos nodos son compensadas con excesos de masa en otros nodos de la propia rejilla.

2). *Afectar la invarianza Galileana.*

Qian y Zhou en 1998 [37] probaron que la invarianza galileana no se pierde si los cambios en la velocidad neta del flujo 'U' en las funciones de distribución de equilibrio f^{eq} son constantes. Esto es, $U + \Delta u = U + cte$.

La matriz de gradientes de velocidad cumple con éste requisito, ya que se puede ver a la matriz como un arreglo de valores constantes que permanecen sin cambio alguno a lo largo de todo el proceso de simulación, por lo que los cambios en las ecuaciones (2.25) y (2.26) respecto a los factores de incremento $\Delta_X(x,y)$ y $\Delta_Y(x,y)$ son constantes en cada instante de tiempo de la simulación y no afectan la invarianza Galileana de la rejilla.

El siguiente segmento de código en Delphi 7.0 muestra como es calculado el valor de las funciones de equilibrio f^{eq} identificadas como 'n_equ' en el código.

```
Procedure CalculaF_Equilibrio;
Var     x,y,i:integer;
        c_squ,t_0,t_1,t_2,u_x,u_y,u_squ,d_loc:real;
        u_n:array[1..8]of real;
        d_loc:real;
Begin
   t_0:=4/9;
   t_1:=1/9;
   t_2:=1/36;
   c_squ:=1/3;
   for x:=1 to lx do
      for y:=1 to ly do
         if (not obst[x,y]) then begin
            d_loc:=0;
            for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y];

            u_x:=(n_hlp[1,x,y]+n_hlp[5,x,y]+n_hlp[8,x,y]
                 -(n_hlp[3,x,y]+n_hlp[6,x,y]+n_hlp[7,x,y]))/d_loc+
                  MatrizDeltas[x,y].X56+ MatrizDeltas[x,y].X13+ MatrizDeltas[x,y].X87;

            u_y:=(n_hlp[2,x,y]+n_hlp[5,x,y]+n_hlp[6,x,y]
                 -(n_hlp[4,x,y]+n_hlp[7,x,y]+n_hlp[8,x,y]))/d_loc+
                  MatrizDeltas[x,y].Y58+ MatrizDeltas[x,y].Y24+ MatrizDeltas[x,y].Y67;

            u_squ:=u_x*u_x+u_y*u_y;

            u_n[1]:=u_x;
            u_n[2]:=u_y;
            u_n[3]:=-u_x;
            u_n[4]:=-u_y;


            u_n[5]:=u_x+u_y;
            u_n[6]:=-u_x+u_y;
            u_n[7]:=-u_x-u_y;
            u_n[8]:=u_x-u_y;

            n_equ[0,x,y]:=t_0*d_loc*(1-u_squ/(2*c_squ));

            n_equ[1,x,y]:=t_1*d_loc*(1+u_n[1]/c_squ
                           +power(u_n[1],2)/(2*power(c_squ,2))
                           -u_squ/(2*c_squ));
            n_equ[2,x,y]:=t_1*d_loc*(1+u_n[2]/c_squ
```

```
                    +power(u_n[2],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
        n_equ[3,x,y]:=t_1*d_loc*(1+u_n[3]/c_squ
                    +power(u_n[3],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
        n_equ[4,x,y]:=t_1*d_loc*(1+u_n[4]/c_squ
                    +power(u_n[4],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));

        n_equ[5,x,y]:=t_2*d_loc*(1+u_n[5]/c_squ
                    +power(u_n[5],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
        n_equ[6,x,y]:=t_2*d_loc*(1+u_n[6]/c_squ
                    +power(u_n[6],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
        n_equ[7,x,y]:=t_2*d_loc*(1+u_n[7]/c_squ
                    +power(u_n[7],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
        n_equ[8,x,y]:=t_2*d_loc*(1+u_n[8]/c_squ
                    +power(u_n[8],2)/(2*power(c_squ,2))
                    -u_squ/(2*c_squ));
    end;
End;
```

Sexto Proceso: El sexto proceso consiste en colisionar y relajar. Para esto se calcula el nuevo valor de las poblaciones f_i con base a la función de distribución de Boltzmann (ver apartado 1.9).

El siguiente segmento de código en Delphi 7.0 muestra como se calcula el nuevo valor de las poblaciones en cada ciclo del sistema.

```
Procedure relaxation(omega:real);
Var     x,y,i:integer;
Begin
   for x:=1 to lx do
      for y:=1 to ly do
         if (not obst[x,y]) then
            for i:=0 to 8 do
                  node[i,x,y]:=n_hlp[i,x,y]+omega*(n_equ[I,x,y]-n_hlp[i,x,y]);
End;
```

Donde N_hlp es una estructura auxiliar en donde se almacenan los valores de la variable 'node' un ciclo antes en el sistema y omega es el parámetro de relajación τ .

Septimo Proceso: El septimo y último proceso es la escribir resultados, el cual consiste en imprimir en pantalla o en impresora los resultados que son producto de la simulación computacional.

2.3.- ANÁLISIS COMPARATIVO DETALLADO .

Análisis del mecanismo de balance de masa :

Para un modelo de rejilla D2Q9, como el ilustrado en la figura 2.12.

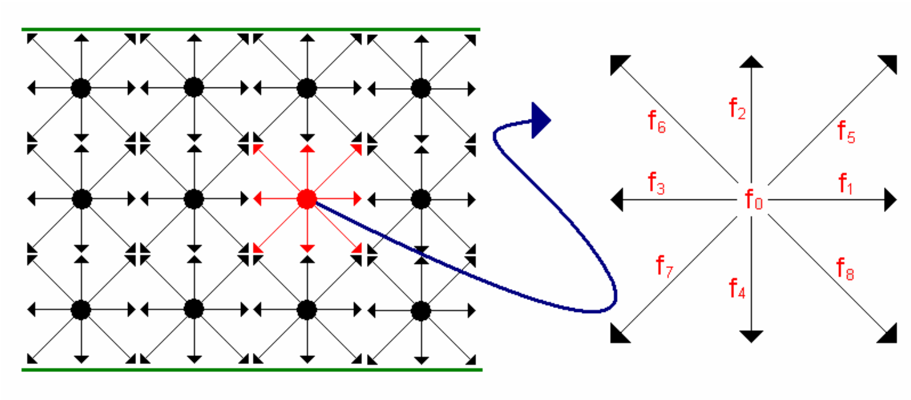


Figura 2.12. Modelo de rejilla D2Q9.

Un valor inicial de las poblaciones f_i tomando a $\rho_0 = 100$ como un valor de densidad local inicial, es el siguiente. (ver figura 2.13)

$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$
$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$
$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$
$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$
$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$
$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$
$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$
$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$	$\frac{100}{9}$	$\frac{400}{9}$	$\frac{100}{9}$
$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$	$\frac{100}{36}$	$\frac{100}{9}$	$\frac{100}{36}$

Figura 2.13. Inicialización de las poblaciones de la rejilla con base al proceso indicado en el apartado 2.2.

Donde el valor de masa global que existe en la rejilla queda determinado por $\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i$.

De esta forma y conociendo el valor de las poblaciones se calcula el valor de masa global en la rejilla: $\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i = 1200$.

Una numeración secuenciada (1..108) de cada una de las poblaciones f_i de la rejilla es mostrada en la figura 2.14.

01	02	03	10	11	12	19	20	21	28	29	30
08	09	04	17	18	13	26	27	22	35	36	31
07	06	05	16	15	14	25	24	23	34	33	32
82	83	84	91	92	93	100	101	102	37	38	39
89	90	85	98	99	94	107	108	103	44	45	40
88	87	86	97	96	95	106	105	104	43	42	41
73	74	75	64	65	66	55	56	57	46	47	48
80	81	76	71	72	67	62	63	58	53	54	49
79	78	77	70	69	68	61	60	59	52	51	50

Figura 2.14. Numeración secuenciada de las poblaciones de la rejilla.

En LBM el valor de masa global debe permanecer estable en cada iteración para asegurar que el sistema converja a una solución en el proceso de simulación. Este principio nos lleva al análisis de la cantidad de masa antes y después del proceso de propagación. El proceso de propagación con el cual se simula en LB el movimiento de las partículas virtuales de fluido (fluores) es realizado de la siguiente forma.
(ver figura 2.15)

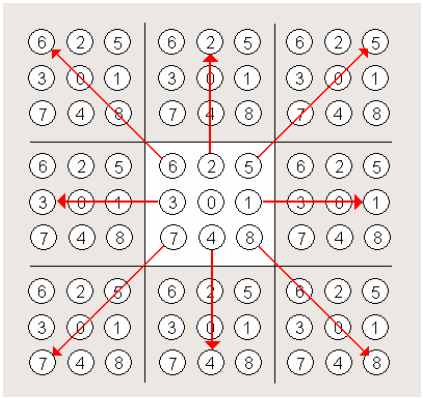


Figura 2.15. Proceso de propagación de las poblaciones f_i para un determinado nodo de la rejilla.

El valor de las poblaciones f_i de todos y cada uno de los nodos que conforman la rejilla es distribuido (desplazado) a diferentes posiciones dentro de los nodos contiguos a cada nodo de la rejilla. (ver figura 2.15)

Para hacer el análisis de la cantidad de masa que existe antes y después del proceso de propagación en la rejilla se partirá del valor de las poblaciones f_i especificados en la figura 2.13.

Para una condición de frontera de tipo periódica, en la que el proceso de propagación toma un comportamiento circular, doblándo a la rejilla en forma imaginaria en el eje 'x' y en el eje 'y', la distribución de las poblaciones f_i después del proceso de propagación y tomando como base a la numeración secuenciada de las poblaciones que se dio, es la siguiente. (ver figura 2.16)

91	83	39	100	92	84	37	101	93	82	38	102
17	09	31	26	18	04	35	27	13	08	36	22
70	78	50	61	69	77	52	60	68	79	51	59
64	74	48	55	65	75	46	56	66	73	47	57
98	90	40	107	99	85	44	108	94	89	45	103
16	06	32	25	15	05	34	24	14	07	33	23
10	02	30	19	11	03	28	20	12	01	29	21
71	81	49	62	72	76	53	63	67	80	54	58
97	87	41	106	96	86	43	105	95	88	42	104

Figura 2.16. Distribución de las poblaciones f_i después del proceso de propagación para una condición de frontera del tipo periódica.

Si se observa éste mismo proceso de propagación con los valores de las poblaciones dados en la figura 2.13, se obtiene la siguiente configuración.
(ver figura 2.17)

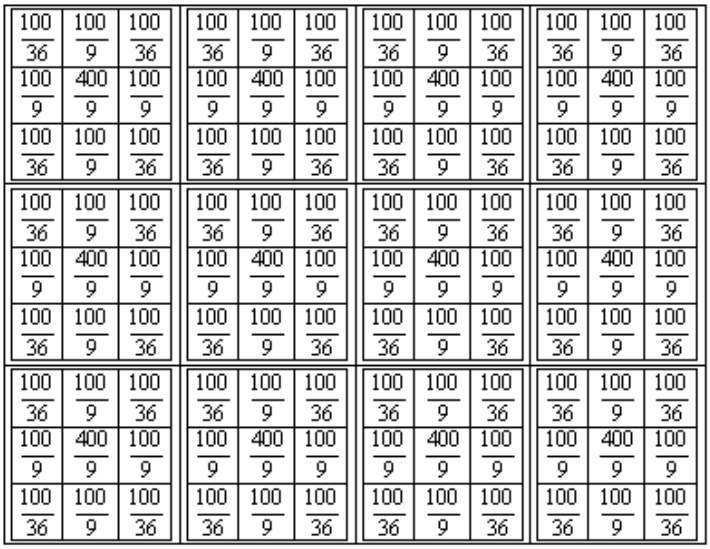


Figura 2.17. Distribución de los valores de las poblaciones después del proceso de propagación bajo una condición de frontera del tipo periódica.

Obsérvese como el movimiento del valor de las poblaciones no afecta el valor de masa global que existía en la rejilla antes del proceso de propagación. De tal forma que se sigue manteniendo constante después de la propagación e igual a $\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i = 1200$.

Ahora bien, para el caso de un mecanismo de fronteras abiertas en el que no existe un doblez imaginario de la rejilla, la distribución de los valores secuenciales de las poblaciones es la siguiente. (ver figura 2.18)

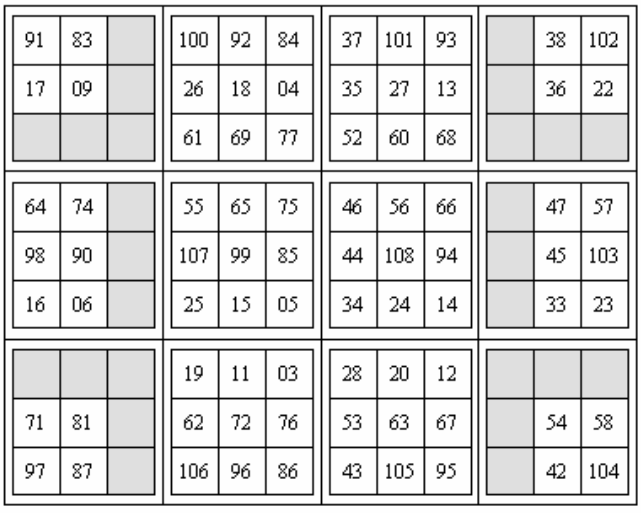


Figura 2.18. Distribución de las poblaciones después del proceso de propagación bajo una condición de fronteras abiertas.

Si se observa éste mismo proceso de propagación pero con los valores de las poblaciones dados en la figura 2.13, se obtienen los siguientes resultados.
(ver figura 2.19)

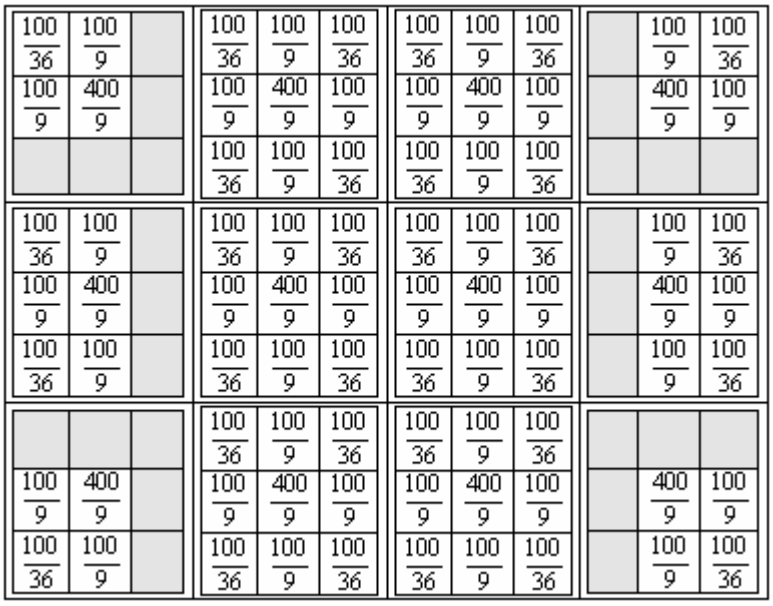


Figura 2.19. Distribución de los valores de las poblaciones después del proceso de propagación bajo una condición de frontera del tipo frontera abierta.

Obsérvese como después del proceso de propagación existen huecos o valores de poblaciones que no pueden ser definidos, ya que no existen nodos contiguos que transfieran valores de población a esos nodos.

Qisu Zou y Xiaoyi He solucionaron el problema definiendo esos valores desconocidos de las poblaciones con base a valores de gradientes de presión y/o velocidad deseados en la entrada y en la salida del ducto, tal como se mostró en el apartado 2.1. Estos gradientes pueden ser manejados como diferencias de densidades en la entrada y salida del ducto. Pero el mecanismo presenta el inconveniente de perder masa en la rejilla después del proceso de propagación.

Para probar como éste mecanismo presenta una varición de masa después del proceso de propagación se calculará con base a las ecuciones (2.5) a (2.10) que proponen Qisu Zou y Xiaoyi y un valor de presión de entrada y salida establecido de antemano como ejemplo, el valor de cada una de las poblaciones f_i cuyo valor es desconocido después del proceso de propagación.

Para esto etiquetemos las poblaciones cuyo valor es desconocido con letras del alfabeto. (ver figura 2.20)

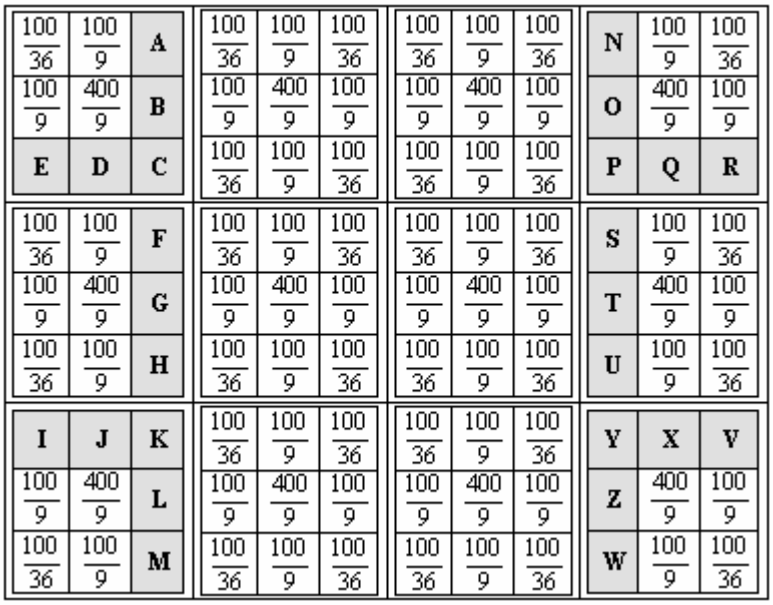


Figura 2.20. Valores desconocidos de poblaciones identificados con letras del alfabeto.

DATOS Y CÁLCULOS

Valor de presión de entrada deseado P=10.

Como el valor de presión esta relacionado con el valor de densidad por medio de la ecuación : $P = C_s^2 \rho$, entonces el valor de densidad para los nodos de entrada es : $\rho_{in} = \frac{P}{C_s^2} = \frac{10}{1/3} = 30$ para una velocidad al cuadrado del sonido de la rejilla (C_S^2) igual a 1/3.

Calculado el valor de la velocidad u_x con base en la ecuación 4.5.

$$u_x = 1 - \frac{\left[f_0 + f_2 + f_4 + 2\left(f_3 + f_6 + f_7\right)\right]}{\rho_{in}} = \frac{\left[\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + 2\left(\frac{100}{9} + \frac{100}{36} + \frac{100}{36}\right)\right]}{30} = 3.33$$

Ahora calculando los valores de las poblaciones f_5, f_1 y f_8 para el nodo central de la entrada (etiquetas F, G y H respectivamente), por medio de las ecuaciones (2.6), obtenemos los siguientes resultados.

(G) $$f_1 = f_3 + \frac{2}{3}\rho_{in}u_x = \frac{100}{9} + \frac{2}{3}(30)(3.33) = 77.71$$

(F) $$f_5 = f_7 - \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x = \frac{100}{36} - \frac{1}{2}\left(\frac{100}{9} - \frac{100}{9}\right) + \frac{1}{6}(30)(3.33) = 19.42$$

(H) $$f_8 = f_6 + \frac{1}{2}(f_2 - f_4) + \frac{1}{6}\rho_{in}u_x = \frac{100}{36} + \frac{1}{2}\left(\frac{100}{9} - \frac{100}{9}\right) + \frac{1}{6}(30)(3.33) = 19.42$$

Por lo que el valor de densidad local calculada para el nodo de entrada es:

$$\sum_{i=0}^{8} f_i = \frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{9} + \frac{100}{36} + 77.71 + 19.42 + 19.42 = 199.88$$

Ahora calculando el valor de las poblaciones f_6, f_2, f_5, f_1 y f_8 (I, J, K, L y M respectivamente) para el nodo de la esquina inferior de la entrada con las ecuaciones (2.7) a (2.10).

(L) $$f_1 = f_3 = \frac{100}{9}$$

(J) $$f_2 = f_4 = \frac{100}{9}$$

(K) $$f_5 = f_7 = \frac{100}{36}$$

(I,M)
$$f_6 = f_8 = \frac{1}{2}\left[\rho_{in} - (f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_7)\right]$$
$$= \frac{1}{2}\left[30 - \left(\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{36}\right)\right] = -32.22$$

El valor de densidad local es:

$$\sum_{i=0}^{8} f_i = \frac{100}{9} + \frac{400}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} - 32.22 - 32.22 = 30$$

Aplicando un procedimiento similar para calcular el valor de las poblaciones f_5, f_1, f_8, f_4 y f_7 (A, B, C, D y E respectivamente) para el nodo de la esquina superior de la entrada, se obtienen los siguientes resultados.

Usando a una variante de las ecuaciones (2.7) a (2.10) se obtiene :

(B) $$f_1 = f_3 = \frac{100}{9}$$

(D) $$f_4 = f_2 = \frac{100}{9}$$

(C) $$f_8 = f_6 = \frac{100}{36}$$

(A,E)
$$f_5 = f_7 = \frac{\rho_{in} - 2f_8 - (f_0 + f_1 + f_2 + f_3 + f_4)}{2}$$
$$= \frac{30 - 2\left(\frac{100}{36}\right) - \left(\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9}\right)}{2} = -32.22$$

El valor de densidad local es:

$$\sum_{i=0}^{8} f_i = \frac{100}{36} + \frac{100}{9} + \frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} - 32.22 - 32.22 = 30$$

Ahora suponiendo un valor de presión de salida deseado de P=5, se calcula el valor desconocido de las poblaciones de los nodos de salida.

Calculando el valor de densidad para los nodos de salida con base al valor de presión deseado: $\rho_{out} = \frac{P}{C_s^2} = \frac{5}{1/3} = 15$

Para calcular el valor de la velocidad u_x se deduce de las ecuaciones (2.2) y (2.3) una ecuación hermana a la ecuación (2.5):

$$f_3 + f_6 + f_7 = \rho_{out} - (f_0 + f_2 + f_4 + f_1 + f_5 + f_8) \quad (2.27)$$

$$f_3 + f_6 + f_7 = f_1 + f_5 + f_8 - \rho_{out} u_x \quad (2.28)$$

De las ecuaciones (2.27) y (2.28) se obtiene:

$$u_x = \frac{\left(f_0 + f_2 + f_4 + 2\left(f_1 + f_5 + f_8\right)\right)}{\rho_{out}} - 1 \tag{2.29}$$

Calculado el valor de la velocidad u_x mediante la ecuación (2.29):

$$u_x = \frac{\left(f_0 + f_2 + f_4 + 2\left(f_1 + f_5 + f_8\right)\right)}{\rho_{out}} - 1 = \frac{\left(\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + 2\left(\frac{100}{9} + \frac{100}{36} + \frac{100}{36}\right)\right)}{15} - 1 = 5.66$$

Ahora calculando los valores de las poblaciones f_6, f_3 y f_7 para el nodo central de la salida (etiquetas S, T y U respectivamente), con base a una variante de las ecuaciones (2.6), obtenemos los siguientes resultados.

(T) $$f_3 = f_1 - \frac{2}{3}\rho_{out}u_x = \frac{100}{9} - \frac{2}{3}(15)(5.66) = -45.48$$

(U) $$f_7 = f_5 + \frac{1}{2}\left(f_2 - f_4\right) - \frac{1}{6}\rho_{out}u_x = \frac{100}{36} + \frac{1}{2}\left(\frac{100}{9} - \frac{100}{9}\right) - \frac{1}{6}(15)(5.66) = -11.37$$

(S) $$f_6 = f_8 - \frac{1}{2}\left(f_2 - f_4\right) - \frac{1}{6}\rho_{out}u_x = \frac{100}{36} + \frac{1}{2}\left(\frac{100}{9} - \frac{100}{9}\right) + \frac{1}{6}(15)(5.66) = 16.92$$

Por lo que el valor de densidad local calculado para el nodo de salida es:

$$\sum_{i=0}^{8} f_i = \frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{9} + \frac{100}{36} - 45.48 - 11.37 + 16.92 = 43.4$$

Ahora calculando el valor de las poblaciones f_6, f_3, f_7, f_4 y f_8 (N, O, P, Q y R respectivamente) para el nodo de la esquina superior de la salida, se obtienen los siguientes resultados.

(O) $$f_3 = f_1 = \frac{100}{9}$$

(Q) $$f_4 = f_2 = \frac{100}{9}$$

(P) $$f_7 = f_5 = \frac{100}{36}$$

(N,R)
$$f_6 = f_8 = \frac{1}{2}\left[\rho_{out} - \left(f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_7\right)\right]$$
$$= \frac{1}{2}\left[15 - \left(\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{36}\right)\right] = -39.72$$

El valor de densidad local es:

$$\sum_{i=0}^{8} f_i = \frac{400}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} - 39.72 = 54.72$$

Calculando el valor de las poblaciones f_5, f_2, f_6, f_3 y f_7 (V, X, Y, Z y W respectivamente) para el nodo de la esquina inferior de la salida, se obtienen los siguientes resultados.

(Z) $$f_3 = f_1 = \frac{100}{9}$$

(X) $$f_2 = f_4 = \frac{100}{9}$$

(Y) $$f_6 = f_8 = \frac{100}{36}$$

(V,W)
$$f_5 = f_7 = \frac{\rho_{out} - 2f_8 - \left(f_0 + f_1 + f_2 + f_3 + f_4\right)}{2}$$
$$= \frac{15 - 2\left(\frac{100}{36}\right) - \left(\frac{400}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9}\right)}{2} = -39.72$$

El valor de densidad local es:

$$\sum_{i=0}^{8} f_i = \frac{400}{9} + \frac{100}{9} + \frac{100}{36} + \frac{100}{9} + \frac{100}{9} + \frac{100}{9} + \frac{100}{36} - 39.72 - 39.72 = 15$$

Entonces, calculando el valor de masa global en la rejilla después del proceso de propagación, tenemos:

$$\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i = 600 + 30 + 199.88 + 30 + 54.72 + 43.4 + 15 = 973$$

Lo que muestra que existe una diferencia de masa en la rejilla de 1200-973=227 después del proceso de propagación.

Otra idea para controlar esa perdida de masa en la rejilla después del proceso de propagación es la mostrada en el apartado 2.2 (cuarto proceso) y forma parte de la contribución original en la creación del nuevo mecanismo de fronteras abiertas propuesto en esta tesis.

Para mostrar como éste mecanismo permite la conservación el total de masa después del proceso de propagación se parte de una rejilla inicializada con valores hipotéticos de densidad local, los cuales pueden ser tomados como ejemplo, los valores secuenciales de las poblaciones f_i mostrados en la figura 2.14.

Los pasos del algoritmo se muestran a continuación:

1). Se calcula la masa global que existe en el sistema antes del proceso de propagación. El cual, para el caso de la rejilla de la figura 2.14, es de $\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i = 5886$.

2). Se propagan los valores de las poblaciones en forma periodica. Lo que nos da como resultado una distribución de rejilla como la mostrada en la figura 2.16.

3). Se sustituyen los valores de las poblaciones f_1, f_5 y f_3 de la primer columna por los valores que tenían éstas poblaciones antes del proceso de propagación, lo que nos da como resultado la siguiente distribución de valores en la primer columna de la rejilla. Ver figura 2.21.

Primer Columna

91	83	03
17	09	04
70	78	05
64	74	84
98	90	85
16	06	86
10	02	75
71	81	76
97	87	77

Figura 2.21. Sustitución de valores en la primer columna de la rejilla.

4). Se almacena el valor de las poblaciones f_i que existe en la última columna de la rejilla en una estructura de nodos euxiliares llamada 'A'. Ver figura 2.22.

Nodos Auxiliares A

82	38	102
08	36	22
79	51	59
73	47	57
89	45	103
07	33	23
01	29	21
80	54	58
88	42	104

Figura 2.22. Estructura de nodos auxiliares 'A'.

5). Se sustituye el valor de las poblaciones f_3, f_6 y f_7 por los valores que tenian éstas antes del proceso de propagación para los nodos de la última columna de la rejilla. Obteniendose una distribución de valores para la última columna como la muestrada en la figura 2.23.

Última Columna

28	38	102
35	36	22
34	51	59
37	47	57
44	45	103
43	33	23
46	29	21
53	54	58
52	42	104

Figura 2.23. Sustitución de los valores anteriores de las poblaciones f_3, f_6 y f_7.

6). Se almacena el valor de las poblaciones f_i que existe en la última columna de la rejilla en una estructura de nodos euxiliares llamada 'B'. Ver figura 2.24.

Nodos Auxiliares B

28	38	102
35	36	22
34	51	59
37	47	57
44	45	103
43	33	23
46	29	21
53	54	58
52	42	104

Figura 2.24. Estructura de nodos auxiliares 'B'.

7). Se calcula la diferencia de masas entre la estructura de nodos auxilares 'A' y la estructura de nodos auxiliares 'B', y el resultado se almacena en una tercer estructura de nodos auxiliares llamada 'Gradientes'. Ver figura 2.25.

Gradientes

54	0	0
-27	0	0
45	0	0
36	0	0
45	0	0
-36	0	0
-45	0	0
27	0	0
36	0	0

=

Nodos Auxiliares A

82	38	102
08	36	22
79	51	59
73	47	57
89	45	103
07	33	23
01	29	21
80	54	58
88	42	104

—

Nodos Auxiliares B

28	38	102
35	36	22
34	51	59
37	47	57
44	45	103
43	33	23
46	29	21
53	54	58
52	42	104

Figura 2.25. Difrencia de masas entre la estructura de nodos auxiliares 'A' y la 'B'.

8). Se retroalimenta por la entrada de la rejilla la masa almacenada en la estructura de nodos auxiliares ‘Gradientes’. Ver figura 2.26.

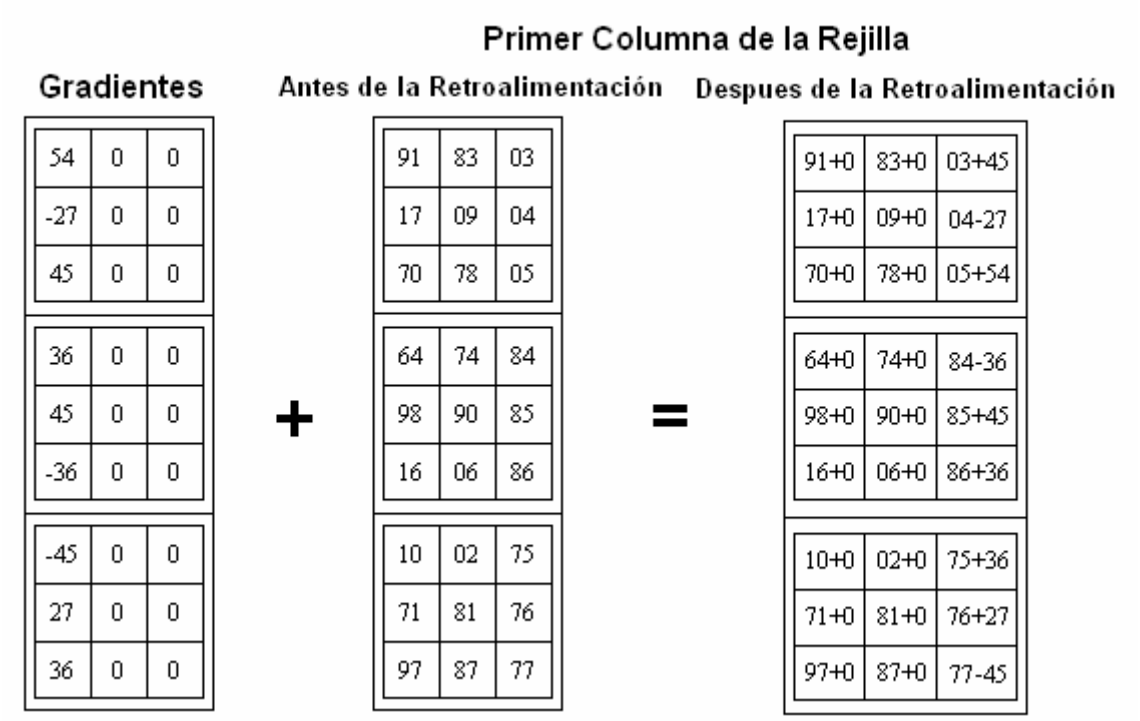


Figura 2.26. Retroalimentación de los valores de la estructura de nodos auxiliares ‘Gradientes’.

9). Finalmente se calcula nuevamente el valor de la masa global que existe en la rejilla después de la retroalimentación. Lo que, para este caso, da como resultado: $\sum_{x=1}^{4}\sum_{y=1}^{3}\sum_{i=0}^{8} f_i = 5886$.

Como se observa este mecanismo conserva la cantidad de masa global que existe antes del proceso de propagación y después de él. Para este caso ejemplo fueron utilizados valores hipotéticos de las poblaciones f_i, pero el mecanismo es tan robusto que trabaja bien para cualquier valor arbitrario de las poblaciones, incluso valores random.

CAPÍTULO III

UN MODELO DE FLUJO SANGUÍNEO EN UNA ARTERIOLA CON REJILLAS DE BOLTZMANN

3.1.- CREACIÓN DE UN MODELO DE ARTERIOLA .

En un artículo publicado en el año 2002 por los investigadores Yasuhiko Sugii, Shigeru Nishio, y Koji Okamoto del laboratorio de investigación en energía nuclear de la universidad de Tokio en Japón [38,39], se publico material bibliografíco referente al comportamiento de la sangre en arteriolas.

Ellos analizaron el sistema cardiocirculatorio de una rata de 8 semanas de nacida mediante la técnica de 'Velocimetría por Imagen de Partículas' (PIV por sus siglas en inglés), y en su artículo denominado: 'Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique' [38], nos muestran un segmento de arteriola en dos dimensiones por el cual circula en tiempo real la sangre de la rata. Cabe destacar que a este nivel dimensional, el diámetro físico de las arteriolas de un ser humano y el de las arteriolas de una rata son similares. Además son igualmente similares los niveles de viscosidad de la sangre de una rata y la de un ser humano.

El artículo es muy valioso para los fines que se persiguen en ésta tesis, ya que además de ilustrar imágenes reales de la geometría de una arteriola, proporciona información y datos acerca de los perfiles de velocidad de la sangre en diferentes puntos de la arteriola y el comportamiento reológico de la sangre en diferentes cortes transversales.

Para analizar el comportamiento del modelo de fronteras abiertas con los datos publicados en este artículo, primero se construyó una imagen discretizada de la arteriola real de la rata. La figura 3.1 muestra la imagen real y la discretizada.

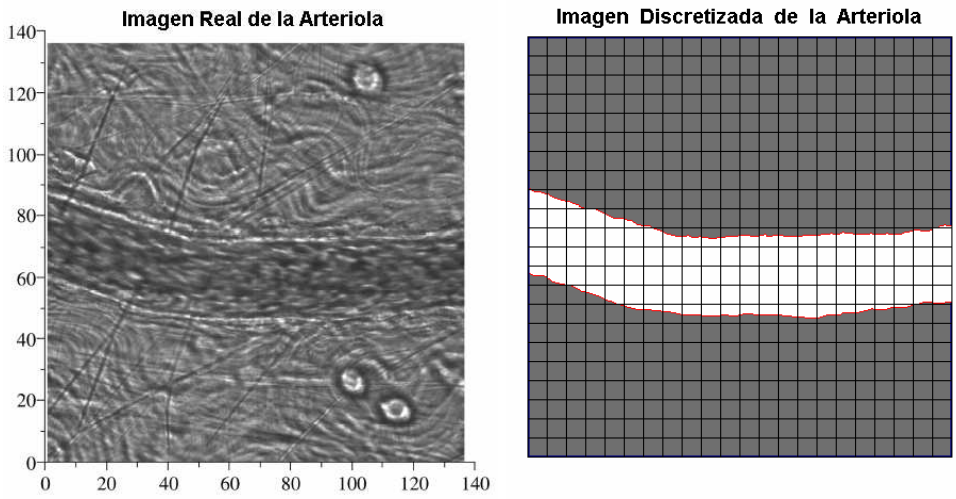


Figura 3.1. Imágenes de la arteriola de una rata .

La imagen discretizada es una replica de la imagen original de la arteriola, ya que al igual que la imagen original que mide 136x136 micrómetros, la imagen discretizada tiene las dimensiones en rejillas de Boltzmann de 136x136 nodos; lo que equivale a un nodo por micrómetro.

Reproducir la geometría original de la arteriola es de suma importancia en el modelo de fronteras abiertas de rejillas de Boltzmann ya que ello nos permite evaluar mediante la simulación, el comportamiento de un fluido de la potencia (sangre) en el interior de un ducto con las mismas dimensiones y geometrías.

Como ya se comentó en apartados anteriores, el modelo de fluidos de la potencia de Oswald nos genera un acercamiento bastante aproximado al comportamiento reológico de la sangre [50], por lo que una vez replicada la geometría original de la arteriola en el modelo de rejillas de Boltzmann, se buscó material bibliográfico que proporcionara datos específicos sobre el índice de consistencia 'k' y el índice de comportamiento del fluido 'n' relacionados con el comportamiento viscoso promedio de la sangre de una rata. Índices que son fundamentales en el modelo de fluidos de la potencia.

En 1999 fue publicado un artículo por los investigadores Mohammad A. Hussain, Subir Kar y Ram R. Puniyani de la Escuela de Ingeniería Biomédica del Instituto de Tecnología de Bombay en la India. Dicho artículo titulado: 'Relationship between power law coefficients and major blood constituents affecting the whole blood viscosity' [50] menciona que se llevó a cabo el análisis del comportamiento viscoso de la sangre de los seres humanos con el fin de obtener el índice de consistencia 'k' y el índice de comportamiento de la sangre 'n' que están relacionados con el modelo de fluidos de la potencia de Oswald. Ellos analizaron el comportamiento reológico de la sangre humana en 79 personas. 43 personas con antecedentes de enfermedades ó padecimientos cardiovasculares y 36 personas completamente sanas. El resultado que ellos publican basado en el análisis del nivel de hematocitos , nivel de fibrinógeno y el nivel de colesterol es que en personas sanas el índice de consistencia promedio es 'k=0.980' y el índice de comportamiento promedio es de 'n=0.708'. Las tablas 3.1 y 3.2 muestran los resultados obtenidos.

Esperando obtener resultados muy cercanos a los reportados por los investigadores de la universidad de Tokio en Japón, fue alimentado el sistema de modelación de la arteriola con la información reportada por los investigadores de la India, debido a la similitud en cuanto al comportamiento viscoso promedio de la sangre de una rata con la de un ser humano.

Los resultados obtenidos del modelo serán una base importante para predecir el comportamiento del esfuerzo de corte y de la velocidad de corte de la sangre humana en arteriolas. Parámetros que son útiles de conocer en la producción de sangre humana artificial [40,41,42,43,44].

Tabla 5.1 Average and standard deviation of calculated *K* of fresh blood samples with respect to its experimental values.

	Average		Difference	
Group	*K* (experimental)	*K* (calculated)	SD	Error (%)
Healthy control (*N*= 36)	0.980	0.995	0.106	9.138
CVA cases (*N*= 43)	1.029	1.054	0.210	15.650

K, Normalized flow consistency index; SD, standard deviation of the difference Between experimental and calculated values.

Tabla 5.2 Average and standard deviation of calculated *n* of fresh blood samples with respect to its experimental values.

	Average		Difference	
Group	*n* (experimental)	*n* (calculated)	SD	Error (%)
Healthy control (*N*= 36)	0.708	0.713	0.025	3.030
CVA cases (*N*= 43)	0.703	0.707	0.027	2.978

n, Non-Newtonian behaviour index; SD, standard deviation of the difference Between experimental and calculated values.

3.2.- EL COMPORTAMIENTO REOLÓGICO DE LA SANGRE .

El fluido sanguíneo es una solución de células y elementos no uniformes suspendidos en un medio acuoso, el plasma. El contenido celular está conformado por eritrocitos, leucocitos y plaquetas. Por su parte, el plasma está conformado en casi su totalidad por agua, y el resto por proteínas y otras sustancias [6].

El componente acuoso le confiere a la sangre un coeficiente de compresión muy alto, por lo que la densidad específica de la sangre se mantiene constante en el rango de presiones fisiológicas. La densidad de la sangre es de 1.1 g/mL, mientras que la del plasma es 1.03 g/mL. [2]

Cuando la sangre fluye por vasos pequeños (microcirculación), por ejemplo, a nivel de los capilares, cuyo diámetro es igual o incluso menor que el de los eritrocitos el flujo no es homogéneo; las células pasan una a una, mientras el plasma fluye entre uno y otro componente celular. En este caso se consideran dos fases: una sólida, dada por las células sanguíneas, y otra líquida, dada por el plasma. Cuando la sangre fluye por vasos sanguíneos amplios (macrocirculación), aunque conserva características de no homogeneidad, porque las células se concentran en el centro de flujo y el plasma fluye por la periferia, al hacer un análisis seccional se puede simplificar el comportamiento como si fuera un fluido uniforme [4].

El análisis del plasma ha mostrado que tiene un comportamiento viscoso constante, por lo que se denomina fluido newtoniano; sin embargo, la sangre completa se comporta como un fluido no newtoniano, en otras palabras, el comportamiento no newtoniano de la sangre se debe a su contenido celular. La flexibilidad eritrocitaria es una propiedad dinámica importante en la mecánica de la circulación sanguínea. Gracias a ella, en la macrocirculación el flujo sanguíneo exhibe un comportamiento que se aproxima al newtoniano, ya que por la forma hidrodinámica que adquiere el eritrocito reduce la perturbación del flujo; mientras que en la microcirculación, se le permite al eritrocito sobrepasar algún tipo de estrechez cambiando su forma [2].

La viscosidad de la sangre varía con tres factores: el hematocrito, la temperatura, y la velocidad de flujo. Como la viscosidad también depende del esfuerzo aplicado (proporcional a la velocidad), se observa que cuando el gradiente de esfuerzos es alto, como el que debe enfrentar en los grandes vasos, la sangre presenta un comportamiento newtoniano. Haciendo una simplificación la sangre se puede considerar entonces como un fluido incompresible, viscoso y homogéneo, cuando se analiza a nivel de macrocirculación [2].

La viscosidad del plasma se acerca a 0.015 Poise (1.5 cP) mientras que la viscosidad de la sangre completa es cercana a los 3.2 centipoise (cP), o $3.2\ 10^{-3}$ Pa.s. [2]

3.3.- LA SANGRE COMO UN FLUIDO DE LA POTENCIA .

A pesar de las recomendaciones por parte del Comité Internacional para la Medida de la Viscosidad de la Sangre Humana (ICSH, 1986), la interpretación y comparación de los datos clínicos de la viscosidad de la sangre en forma completa no son del todo satisfactorios [45]. Esto es porque los investigadores en este campo generalmente reportan valores de viscosidad en los cuales usan uno o dos valores particulares de deformación angular diferentes. (Walker *en* 1985; Lechner *en* 1986; Ott *en* 1986; Caimi *en* 1987; Ernst *en* 1987; y Fisher *en* 1987).

Charm y Kurland (1962) sugirieron que la relación entre la deformación angular y viscosidad aparente de la sangre sigue la ley de la potencia. Esto fue también publicado por Benis en 1971. Bernasconi en 1989 y Kar en 1991 los cuales demostraron que el método de regresión en la ley de la potencia es adecuado para cuantificar la viscosidad de la sangre. Bernasconi en 1991 condujo una serie de experimentos exclusivamente tomando muestras de sangre de pacientes sanos, mientras que Kar en 1991 y later Hussain en 1994, del mismo grupo de investigadores probaron concluyentemente que no solo la sangre de pacientes sanos, sino que la sangre proveniente de pacientes con patologías también sigue el modelo de la ley de la potencia en su comportamiento. Los resultados obtenidos son ampliamente usados para entender el comportamiento reológico de la sangre [45].

Puesto que ‘n’ del modelo de la ley de la potencia es el índice de comportamiento no Newtoniano y ‘k’ es el índice de consistencia del flujo, ellos deben ser naturalmente dependientes de ciertos constituyentes de la sangre tales como el nivel de hematocitos , el nivel de fibrinógeno, y nivel de colesterol, pero existen pocos estudios en la actualidad en los cuales se relaciona los coeficientes de la ley de la potencia con el nivel de hematocitos y parámetros bioquímicos que constituyen a la sangre [45].

3.4.- EL MODELO DE FLUIDO DE LA POTENCIA USADO EN EL MODELO DE ARTERIOLA .

En el modelo de arteriola se implemento el modelo de fluidos de la potencia de Oswald, el cual, como ya se comentó en el apartado 1.10, usa dos parámetros que deben ser conocidos de antemano: el índice de comportamiento y el índice de consistencia del fluido. Con estos dos parámetros y conociendo los gradientes de la velocidad del flujo en el interior del ducto se puede conocer el comportamiento reológico del fluido.

El índice de consistencia del fluido (*k*) y el índice de comportamiento del fluido (*n*) son plenamente conocidos mediante la información desplegada en las tablas 5.1 y 5.2 respectivamente. Y los gradientes de velocidad del flujo se conocen en cada instante de tiempo ‘t’ y en cada punto (x,y) de la rejilla. Por lo que una vez implementado el modelo de fluido de la potencia se puede analizar el comportamiento del modelo de arteriola bajo diferentes valores de ‘k’ y de ‘n’.

3.5.- DESCRIPCIÓN DEL MODELO DE ARTERIOLA .

El modelo de arteriola esta implementado en lenguaje Delphi versión 7.0. Consta de 11,000 líneas de programación y corre en un sistema operativo Windows 2000 o XP bajo una plataforma Pentium 4 con un requerimiento de memoria mínima de 256 Mb.

El sistema es completamente visual y consta de 4 módulos. El primero de ellos tiene como objetivo mostrar la imagen original de la arteriola tomada del artículo 'Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique' [56], y de la cual fueron extraídos los datos relacionados con la geometría y dimensiones reales de la arteriola. (ver figura 3.2).

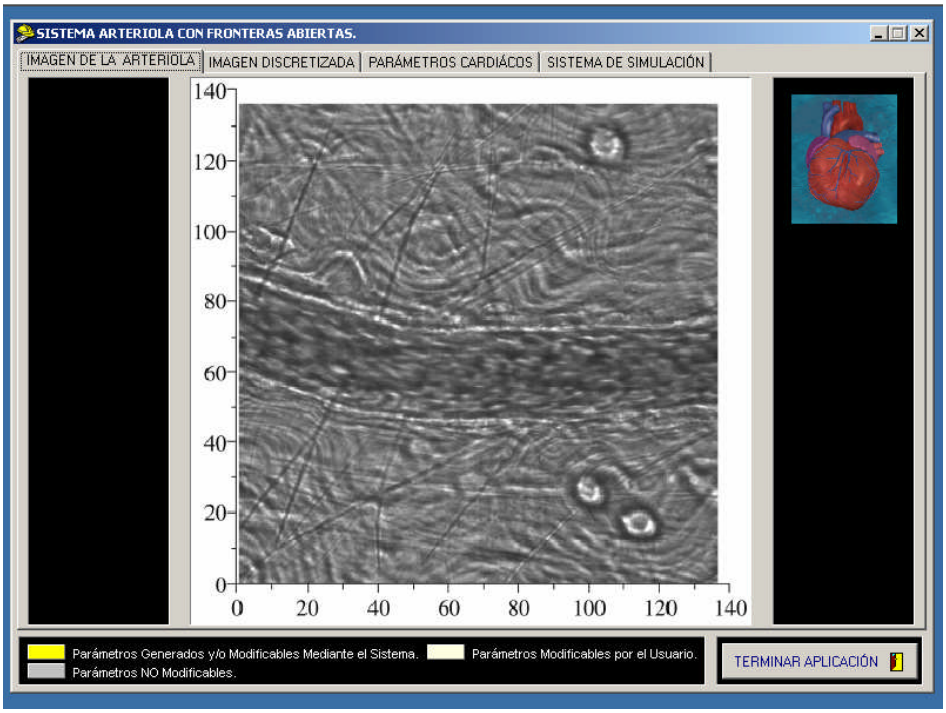


Figura 3.2. Imagen correspondiente del módulo I del modelo de arteriola. En ella se aprecia la Imagen real de la arteriola de una rata de laboratorio.

El segundo módulo permite ver la imagen discretizada de la arteriola real al igual que diversas opciones entre las que destacan: cambiar punto de corte y ver flujo (ver figura 3.3). La opción <<cambiar punto de corte>> permite al usuario mover una línea negra vertical ubicada en la imagen discretizada de la arteriola que indica la línea transversal en la cual se desea hacer un análisis de los perfiles de velocidad del fluido en cuestión (sangre).

La opción <<ver flujo>> da la posibilidad de ver el comportamiento del flujo del fluido a lo largo de la arteriola. La figura 3.4 muestra la imagen del comportamiento del fluido, la cual permite observar las direcciones que toma el flujo en el interior del ducto.

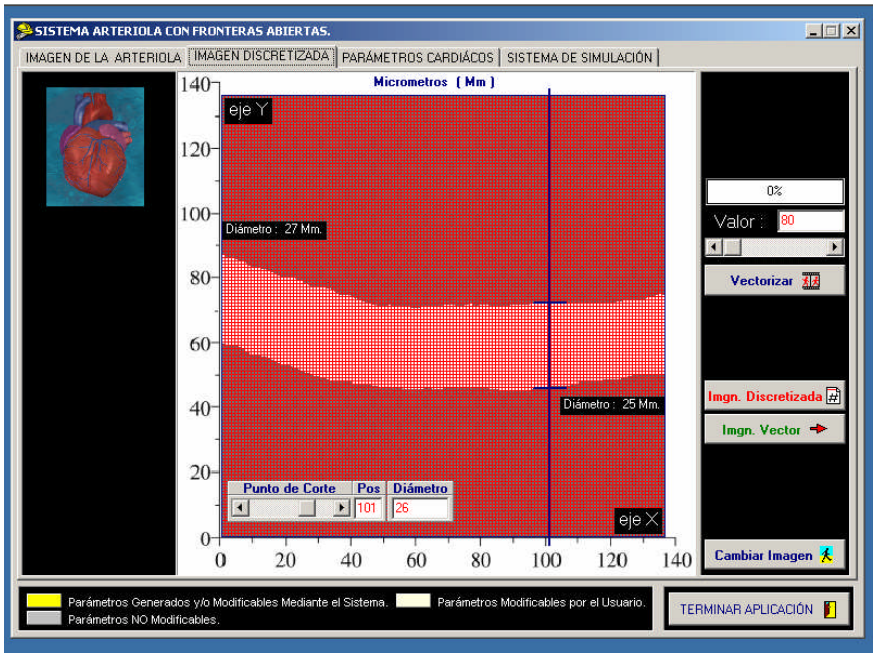


Figura 3.3. Imagen discretizada de la arteriola que muestra el punto de corte como una línea vertical.

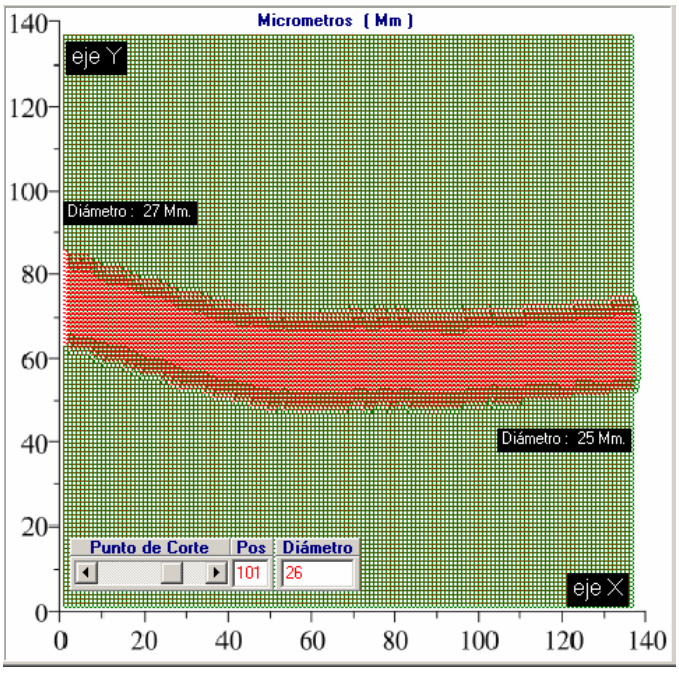


Figura 3.4. Imagen vectorizada del flujo de la sangre en el interior de la arteriola.

El tercer módulo tiene la finalidad de cambiar ciertos parámetros en la simulación, tales como el comportamiento reológico del fluido, esto es, si se esta simulando el comportamiento de un fluido 'Newtoniano' o el comportamiento de uno 'No Newtoniano'. También permite cambiar el perfil de velocidades si se trata de un fluido 'No Newtoniano' y la velocidad de entrada del fluido en el ducto. Los parámetros reológicos 'n' y 'k' para el caso de fluidos No Newtonianos también pueden ser establecidos en este apartado. La figura 5.5 muestra una imagen de este subsistema.

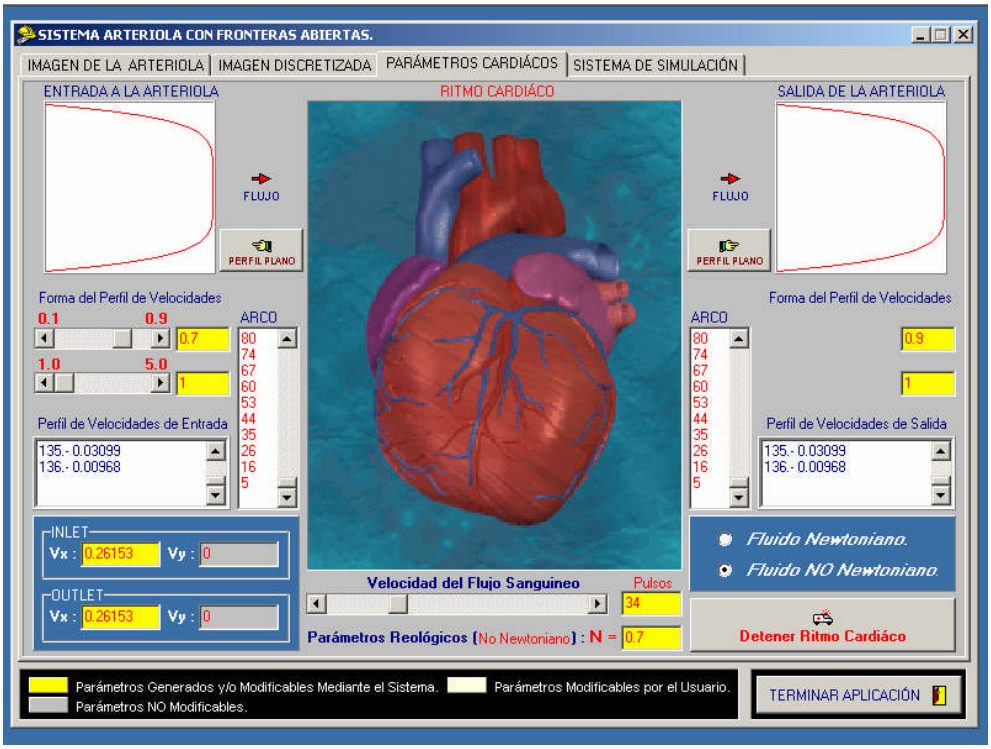


Figura 3.5. Tercer módulo del sistema que permite cambiar parámetros relevantes para indicar el tipo de fluido que se desea simular.

El cuarto y último módulo permite ejecutar el proceso de simulación y nos muestra en cada iteración del sistema el perfil de velocidades que se esta obteniendo en el punto de corte especificado previamente. También muestra el comportamiento reológico del fluido y las velocidades y viscosidades promedio a lo largo de la arteriola. La figura 3.6 muestra este subsistema.

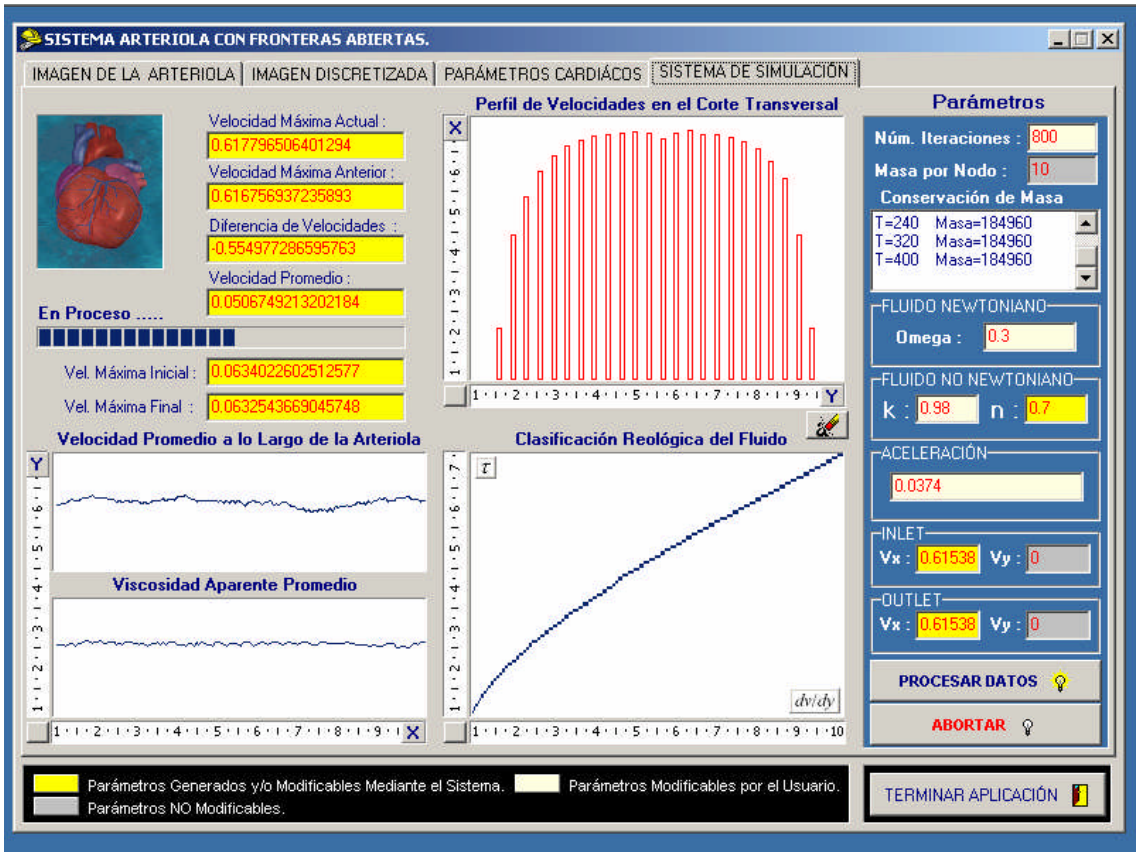


Figura 3.6. Cuarto y último módulo el cual muestra los resultados de la simulación.

3.6.- DIFICULTADES TÉCNICAS EN LA IMPLEMENTACIÓN COMPUTACIONAL DEL MODELO DE ARTERIOLA Y SU SOLUCIÓN .

Problema (1) : El GRADIENTE DE VELOCIDAD EN EL MODELO DE FLUIDOS DE LA POTENCIA.

De acuerdo con el segundo invariante ‘e’ de la velocidad del tensor de deformación, especificado en la ecuación (1.70) (ver apartado 1.10), en el cual se relacionan los gradientes de velocidad del flujo en ‘x’ y en ‘y’, se tiene que para un gradiente de velocidad , por ejemplo en x :

$$\frac{\partial u_x}{\partial x} \tag{3.1}$$

su solución discreta mediante diferencias divididas finitas centradas [11,57] es :

$$\frac{\partial u_x}{\partial x} = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12h} \tag{3.2}$$

donde u_i denota el valor de ‘ u_x ’ en la posición ‘ i ’ y h es el tamaño de peso que establece el valor de separación entre un valor de muestra u_i y su valor sucesor u_{i+1} o su valor predecesor u_{i-1}. Para este caso h=1, debido a que la separación entre velocidades u_i en la rejilla discreta es de 1 nodo.

El método de diferencias divididas finitas centradas prersenta un error estimado del 0%. A diferencia del de diferencias hacia adelante y hacia atrás, que presentan un error estimado del 5.8% y 3.7% respectivamente. Esto es debido a que a que las fórmulas se basan en la serie de Taylor, y son equivalentes a polinomios que pasan a través de los puntos.

La solución a la ecuación (5.2) es simple cuando la rejilla de Boltzmann representa un ducto recto y uniforme, ya que la dirección del flujo siempre será en línea recta y u_i y u_{i+1} siempre estarán alineados en forma perpendicular a dicha línea recta. (ver figura 3.7)

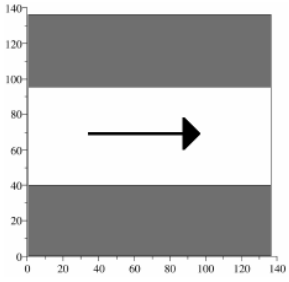


Figura 3.7. Ducto recto con geometría regular.

Cuando se trata de un ducto irregular en el que tanto su geometría como su inclinación local varia, como es el caso de la arteriola (ver figura 3.8). La dirección del flujo es variable y por lo tanto las posiciones de las velocidades u_i y u_{i+1}pueden estar localizadas en puntos diferentes a una alineación vertical.

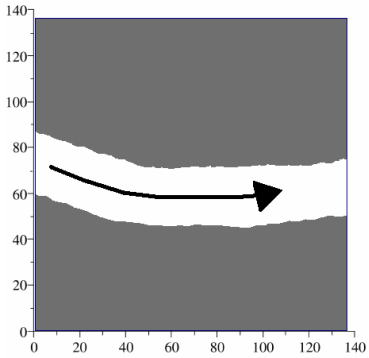


Figura 3.8. Ducto con geometría e inclinación local irregular.

Solución :

Para solucionar este problema y poder localizar adecuadamente en la rejilla estos puntos en los cuales se encuentran las velocidades u_i y u_{i+1}, se calcula una traza central la cual es una línea curva que atraviesa el ducto por su centro. (ver figura 3.9)

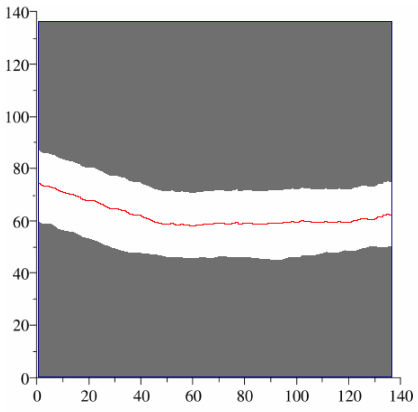


Figura 3.9. Ducto con traza central.

Donde las coordenadas (x_{traza} , y_{traza}) de los puntos que integran la traza central son calculados de la siguiente manera :

$$1 \leq X_{Traza} \leq Lx \quad ; \quad y_{Traza} = \frac{y_{Punto\ Superior} - y_{Punto\ Inferior}}{2} \tag{3.3}$$

Donde ' $y_{Punto\ Superior}$' y '$y_{Punto\ Inferior}$' son los valores de las coordenadas en el eje 'y' del par de puntos perpendiculares que integran las paredes superior e inferior del ducto respectivamente en una determinada posición en 'x'. Y 'Lx' representa el límite de la retícula en el eje 'x'.

Para determinar la recta perpendicular a la traza central en la cual se encuentran localizadas las velocidades u_i y u_{i+1}, se calcula la recta tangente a la traza central en cada punto que la integra. Para esto se determina la pendiente a cada punto usando la ecuación de la pendiente :

$$m_T = \frac{Y_2 - Y_1}{X_2 - X_1} \tag{3.4}$$

y la pendiente de la recta perpendicular a la tangente es determinada por :

$$m_P = -\frac{1}{m_T} \tag{3.5}$$

Donde (x_1,y_1) y (x_2,y_2) representan las coordenadas de dos puntos contiguos en la traza central. Y con ' m_p ' se calculan los puntos que integran la recta perpendicular a la recta tangente en cada punto de la traza central.

Problema (2) : LA MATRIZ DE GRADIENTES DE VELOCIDAD EN EL MODELO DE FRONTERAS ABIERTAS DE REJILLAS DE BOLTZMANN.

(A)

Debido a que la correspondencia de puntos que integran la matriz de gradientes de velocidad y los que conforman la retícula de Boltzmann en principio es uno a uno, la funcionalidad de la aplicación de la matriz se limita solo a aquellos casos en donde las dimensiones de la retícula de Boltzmann ocupa la totalidad del diámetro del ducto por el cual fluye el fluido. Para aquellos casos en donde el diámetro es inferior, como es el caso de la arteriola, se debe hacer un ajuste para adaptar la matriz de gradientes de velocidad solo a los nodos que integran el ducto en la retícula. (ver figura 3.10)

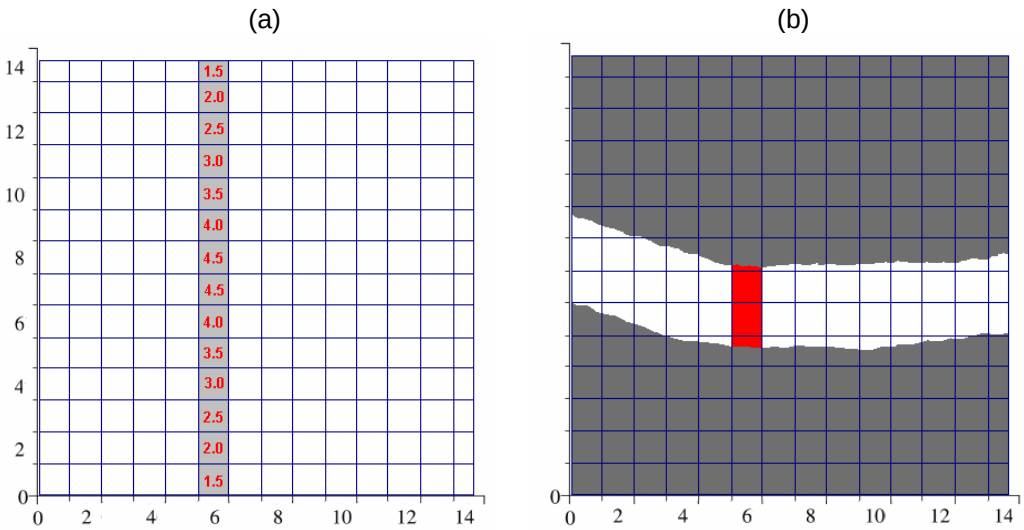


Figura 3.10. (a) **Matriz de gradientes de velocidad. (b) Imagen discretizada de la arteriola.**

Solución :

Para estos casos es conveniente hacer un ajuste de escala mediante una selección de coordenadas (x,y) en la matriz de gradientes que correspondan a las coordenadas específicas que conforman el ducto de la arteriola. Para esto puede ser aplicado cualquier método de ajuste de proporción conocido.

(B)

La irregularidad en la geometría del ducto es también un problema para el ajuste de la matriz de gradientes de velocidad. (ver figura 3.11)

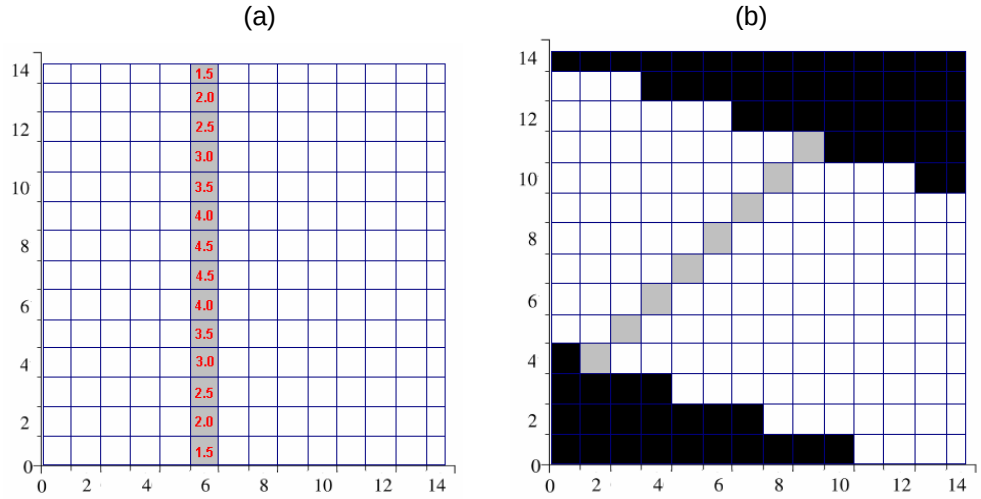


Figura 3.11. (a) **Matriz de gradientes de velocidad.** **(b) Ducto con geometría irregular.**

Solución :

Para la solución a este problema es de gran ayuda combinar el concepto de ajuste de proporción y el concepto de traza central descrito con anterioridad. La combinación de ambos conceptos permite ajustar en proporción y en dirección la matriz de gradientes de velocidad a los puntos que conforman el ducto en la retícula de Boltzmann.

CAPÍTULO IV

RESULTADOS OBTENIDOS CON EL MODELO DE ARTERIOLA

4.1.- ANÁLISIS DE UNIDADES EN REJILLAS DE BOLTZMANN .

Dado que todas las unidades que se manejan en una rejilla de Boltzmann son adimensionales es necesario hacer una conversión del espacio físico a unidades adimensionales y posteriormente hacer una conversión de los resultados de las simulaciones a unidades físicas para poder cuantificarlos.

En este apartado se muestra los pasos básicos elementales para hacer tales conversiones usando la manera de trabajar de Succi [17].

- **Longitud :**

En la mayoría de las simulaciones de LBE se asume que el espacio de rejilla Δx es la unidad de espacio [17], por lo tanto, usando N puntos de la cuadricula para representar un tamaño lineal de L centímetros, la unidad de espacio es :

$$S_L \equiv \Delta x = \frac{L}{N} \text{ (cm)} \tag{4.1}$$

De igual forma para Δy [17].

Análisis Particular :

Para el caso particular de nuestro modelo de arteriola se tiene que se uso una rejilla cuadrada de 136x136 nodos para representar una imagen real de 136x136 Micrómetros. Esto nos da una proporción 1:1, lo que equivale a 1 unidad de red (nodo) por Micrómetro.

- **Velocidad :**

Es común usar en rejillas de Boltzmann la velocidad el sonido C_S en el medio como punto de referencia para dimensionar las velocidades que son resultado de las simulaciones (V_R). Por lo que para convertir las unidades físicas de velocidad a unidades adimensionales de velocidad para un modelo de rejilla de Boltzmann (V_A) se usa el número de Mach, el cual es la razón entre una velocidad física determinada V_R y la velocidad del sonido C_S en el medio.

$$V_R = (V_A)(C_S) \tag{4.2}$$

$$V_A = \frac{V_R}{C_S} \tag{4.3}$$

Análisis Particular:

Para el caso particular de nuestro modelo de arteriola basado en un esquema de fronteras abiertas de rejillas de Boltzmann en el cual se manejan velocidades deseadas de entrada y de salida, las conversiones de velocidad son necesarias.

A nivel de ejemplo, porque es uno de mis casos, tomemos los siguientes datos:

Velocidad de entrada deseada = 3 mm/seg.

Velocidad de salida deseada = 5 mm/seg.

Velocidad del sonido en la sangre [6] = 1570×10^3 mm/seg.

Velocidad adimensional resultado de la simulación = 2.547×10^{-6}

Haciendo la conversión a unidades de velocidad adimensionales se tiene:

Número de Mach de entrada deseado = $\frac{V_R}{C_S}$

$$= \frac{3 \text{ mm/seg}}{1570 \text{x} 10^3 \text{ mm/seg}}$$

$= 1.91 \times 10^{-6}$ adimensional

Número de Mach de salida deseado = $\frac{V_R}{C_S}$

$$= \frac{5 \text{ mm/seg}}{1570 \text{x} 10^3 \text{ mm/seg}}$$

$= 3.184 \times 10^{-6}$ adimensional

Y haciendo la conversión a unidades dimensionales se tiene:

Velocidad real = $(V_A)(C_S)$ = $(2.547 \times 10^{-6}) (1570 \times 10^3 \text{ mm/seg})$

= 3.998 mm/seg.

4.2.- CÁLCULO DEL NÚMERO DE REYNOLDS Y VALORES DE PRESIÓN.

Se tiene que el número de Reynolds para un fluido de la potencia [46] esta dado por:

$$Re = \frac{8\ (V)^{2-n}\ R^{n}\ \rho}{k\left[\frac{3n+1}{n}\right]^{n}} \qquad (4.4)$$

Y la caída de presión por fricción, según la ecuación de Fanning para fluidos no Newtonianos [65] está dado por:

$$\Delta P = \frac{2LK(V)^{n}}{(R)^{n+1}}\left[\frac{3n+1}{n}\right]^{n} \qquad (4.5)$$

Donde 'R' es el radio del ducto, 'V' es la velocidad promedio del fluido, ' ρ ' es la densidad específica del fluido y 'k' y 'n' son los índices de consistencia y de comportamiento del fluido respectivamente.

A). ***Cálculo del número de Reynolds experimental.***

Del artículo : “Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique “ [38] se tiene que la máxima velocidad promedio alcanzada de la sangre en la arteriola, dada una velocidad constante de entrada de 2.7 mm/seg y una velocidad constante de salida de 3.6 mm/seg, es de 3.35 mm/seg.

Además se cuenta con los siguientes datos experimentales:

- Diámetro de la arteriola [56]: 23.6 Micrometros = $23.6\text{x}10^{-6}$ metros
- Densidad específica de la sangre humana [8]: 0.0011 g/mm^3 = 1100 kg/mto^3
- Indice de comportamiento experimental promedio de la sangre (n) adimensional [50]: 0.708
- Indice de consistencia experimental promedio de la sangre humana (k) dimensional [50]: 16.66 mPa.seg ^{n} =$16.66\text{x}10^{-3}$ Pa.seg ^{n}

Con estos datos experimentales y usando la ecuación (4.4) se calcula el número de Reynolds experimental:

$$\text{Re}_{\text{Experimental}} = \frac{(8)\left(3.35\text{x}10^{-3}\right)^{(2-0.708)}\left(\frac{23.6\text{x}10^{-6}}{2}\right)^{0.708}(1100)}{\left(16.66\text{x}10^{-3}\right)\left[\frac{(3)(0.708)+1}{(0.708)}\right]^{(0.708)}} = 3.798\text{x}10^{-2}$$

B). ***Cálculo del valor de presión experimental.***

Usando los datos experimentales del punto (A) y la ecuación (4.5) obtenemos un valor dimensional para la caída de presión:

$$\Delta P_{\text{Experimental}} = \frac{2\left(136\text{x}10^{-6}\right)\left(16.66\text{x}10^{-3}\right)\left(3.35\text{x}10^{-3}\right)^{0.708}}{\left(\frac{23.6\text{x}10^{-6}}{2}\right)^{0.708+1}}\left[\frac{(3)(0.708)+1}{(0.708)}\right]^{0.708} = 59.92 \text{ Pa}$$

C). ***Cálculo del número de Reynolds del modelo de arteriola.***

Para calcular el número de Reynolds del modelo de arteriola se deben de tomar todos los valores en forma adimensional. Por lo que se tiene que:

- Diámetro de la arteriola: 24 unidades de red
- Indice de comportamiento experimental promedio de la sangre (n) adimensional [50]: 0.708
- Indice de consistencia (k) adimensional: 2.08×10^{-5}
- Velocidad adimensional promedio en el modelo: 2.038×10^{-6}
- Conversión del valor dimensional de la densidad de la sangre a valor adimensional usando el valor de la densidad del agua:

  (a) densidad del agua [8] = 1000 kg/mto^3

  (b) densidad de la sangre [8] = 1100 kg/mto^3

  Valor adimensional de la densidad = $\dfrac{1100 \text{ kg/mto}^3}{1000 \text{ kg/mto}^3} = 1.1$

Cálculo del número de Reynolds para el modelo de arteriola:

$$\text{Re}_{Modelo} = \frac{(8)\left(2.038 \times 10^{-6}\right)^{(2-0.708)} \left(\dfrac{24}{2}\right)^{0.708} (1.1)}{\left(2.08 \times 10^{-5}\right) \left[\dfrac{(3)(0.708)+1}{(0.708)}\right]^{(0.708)}} = 3.798 \text{x} 10^{-2}$$

El valor adimensional del índice de consistencia (k) fue calculado de antemano para hacer que ambos numeros de Reynolds (el experimental y el del modelo) converjan en un mismo valor.

D). *Cálculo del valor de presión del modelo .*

Usando los datos adimensionales obtenidos en el punto (C), el valor adimensional del indice de consistencia 'k' (2.08×10^{-5}) calculado previamente para hacer que ambos números de Reynolds (el experimental y el del modelo) converjan en un mismo valor y usando la ecuación (4.5) se obtiene un valor adimensional de la caída de presión en el modelo de arteriola:

$$\Delta P_{Modelo} = \frac{2(136)(2.08x10^{-5})(2.038x10^{-6})^{0.708}}{\left(\frac{24}{2}\right)^{0.708+1}}\left[\frac{(3)(0.708)+1}{(0.708)}\right]^{0.708} = 2.1713x10^{-8}$$

Convirtiendo el valor de presión adimensional a valor dimensional, se tiene:

$$P_{Modelo} = P^{*}C_S^2\ \rho = (2.1713x10^{-8})\ (1570)^2(1100) = 58.875\ Pa$$

Donde P^{*} es el valor adimensional de la caída de presión ($2.171x10^{-8}$), C_S es la velocidad real del sonido en la sangre (1570 mts/seg) y 'ρ' es el valor dimensional de la densidad específica de la sangre (1100 kg/mto^3).

Esto muestra que el valor de presión manejado por el modelo de arteriola es congruente con el valor real calculado en el punto (B).

4.3.- VALIDACIÓN DEL MODELO DE ARTERIOLA CON DATOS EXPERIMENTALES.

Para la validación del modelo de arteriola se tomaron los datos experimentales publicados en el artículo 'Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique' referenciado previamente [38]. En él se especifican valores observados de velocidad máxima alcanzada en la entrada y en una longitud específica de 85 Mm de la arteriola. Estas velocidades son asociadas a un valor específico de tiempo que marca el ritmo de los ciclos sístole y diástole del corazón (ver tabla 4.1). La figura 4.1 muestra los perfiles de velocidad de la sangre en cada uno de estos tiempos.

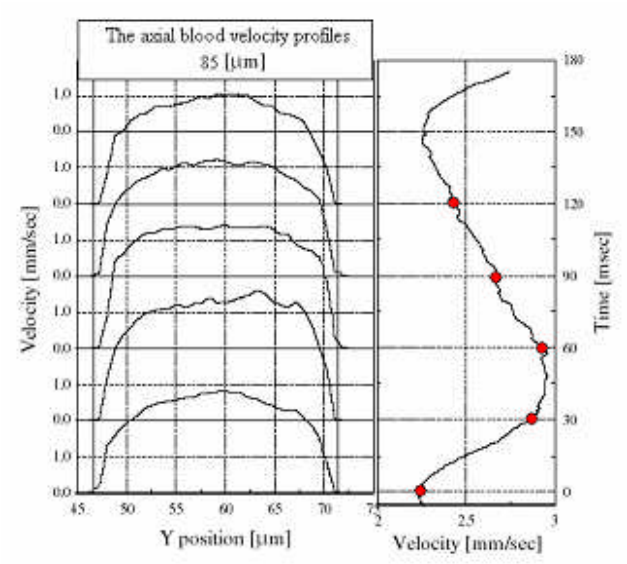


Figura 4.1. Perfiles de velocidad en los ciclos sístole y diástole del corazón de una rata de laboratorio.

Tiempo (mseg)	Velocidad de Entrada (mm/seg)	Velocidad Máxima en 85 Micrómetros
0	2.2	2.8
30	2.8	3.5
60	2.9	3.6
90	2.7	3.3
120	2.4	3.0

Tabla 4.1. Velocidades asociadas con el ciclo sístole y diástole del corazón de una rata de laboratorio.

Con los valores adimensionales del índice de consistencia $k=2.08x10^{-5}$ e índice de comportamiento del fluido n=0.708 manejados en el apartado 4.2 inciso (C) y con los valores de velocidad de entrada especificados en la tabla (6.1) dadas en forma adimensional se alimentó el modelo de arteriola para obtener el valor máximo de velocidad en 85 Micrometros y de esta manera comparar los resultados que se obtienen vía modelo con los valores experimentales publicados en la tabla (4.1).

La conversión de velocidades de entrada dimensionales a valores adimensionales con que es alimentado el modelo (número de Mach) son expresados en la tabla 4.2.

Tiempo (mseg)	Velocidad de Entrada (mm/seg)	Conversión de la Velocidad de Entrada a Valor Adimensional	Velocidad Adimensional (Número de Mach)
0	2.2	$2.2 / 1570x10^{3}$	$1.40x10^{-6}$
30	2.8	$2.8 / 1570x10^{3}$	$1.78x10^{-6}$
60	2.9	$2.9 / 1570x10^{3}$	$1.84x10^{-6}$
90	2.7	$2.7 / 1570x10^{3}$	$1.72x10^{-6}$
120	2.4	$2.4 / 1570x10^{3}$	$1.53x10^{-6}$

Tabla 4.2. Valores adimensionales de las velocidades de entrada.

Los resultados estables de las simulaciones después de 150,000 iteraciones muestran que para éstas velocidades adimensionales de entrada se generan valores de velocidad adimensional máximos en 85 Micrometros de $1.719x10^{-6}$, $2.293x10^{-6}$, $2.420x10^{-6}$, $2.038x10^{-6}$ y $1.847x10^{-6}$, los cuales convirtiendos a unidades físicas nos dan valores de 2.7, 3.6, 3.8, 3.2 y 2.9 mm/seg respectivamente. La tabla 4.3 muestra los valores adimensionales de velocidad del modelo y su conversión a unidades físicas.

Velocidad máxima adimensional obtenida en 85 Micrómetros	Velocidad en unidades físicas (mm/seg)
$1.719x10^{-6}$	2.7
$2.293x10^{-6}$	3.6
$2.420x10^{-6}$	3.8
$2.038x10^{-6}$	3.2
$1.847x10^{-6}$	2.9

Tabla 4.3. Valores dimensionales de las velocidades en 85 Micrómetros.

Con estos valores se genera una gráfica comparativa, la cual muestra que ambos grupos de datos se comportan con la misma tendencia. (ver figura 4.2)

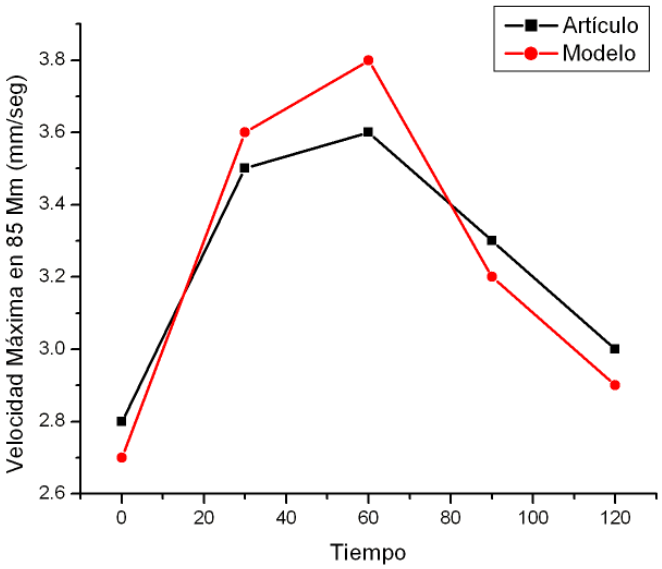


Figura 4.2. Gráfica comparativa.

El artículo también proporciona datos experimentales relacionados con el valor de la velocidad de la sangre en diferentes puntos a lo largo de la arteriola real con una velocidad constante de entrada de 2.7 mm/seg y una velocidad constante de salida de 3.5 mm/seg. Estos puntos son : 53, 69, 85, 101, 117 y 125 Micrómetros. La tabla 4.4 muestra los datos publicados en el artículo.

Posición en la Arteriola (Micrómetros)	Diámetro de la Arteriola (Micrómetros)	Velocidad Máxima Experimental (mm/seg)
53	24.9	3.3
69	24.9	3.3
85	24.7	3.3
101	25.7	3.1
117	24.1	3.5
125	23.6	3.6

Tabla 4.4. Valores de velocidad en diferentes puntos a lo largo de la arteriola en condiciones constantes de velocidad de entrada y de salida.

El modelo fue alimentado con estas velocidades de entrada y de salida constantes de 2.7 y 3.5 mm/seg, las cuales convertidas a valores adimensionales se obtiene: $1.719x10^{-6}$ y $2.229x10^{-6}$ respectivamente. Los resultados estables de las simulaciones muestran valores adimensionales de velocidad máxima alcanzados en éstos puntos de: $1.974x10^{-6}$, $2.165x10^{-6}$, $2.133x10^{-6}$, $1.942x10^{-6}$, $2.261x10^{-6}$ y $2.356x10^{-6}$, los cuales convertidos a unidades físicas de velocidad se obtiene: 3.10, 3.40, 3.35, 3.05, 3.55 y 3.70 respectivamente. La tabla 4.5 muestra éstos resultados.

Posición en la Arteriola (Micrómetros)	Velocidad Máxima EXPERIMENTAL (mm/seg)	Velocidad Máxima MODELO (mm/seg)
53	3.3	3.10
69	3.3	3.40
85	3.3	3.35
101	3.1	3.05
117	3.5	3.55
125	3.6	3.70

Tabla 4.5. Valores de velocidad máxima obtenidos en el modelo.

La gráfica comparativa de la figura 4.3 muestra el resultado de las tendencias entre ambos grupos de datos.

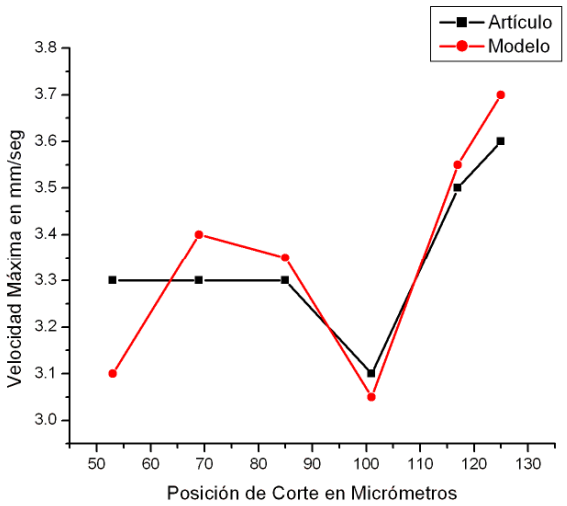


Figura 4.3. Gráfica comparativa .

La figura 4.4 muestra un análisis comparativo gráfico entre los perfiles de velocidad observados experimentalmente en diferentes puntos a lo largo de la arteriola: 53 Mm, 69 Mm, 85 Mm, 101 Mm, 117 Mm y 125 Mm, los cuales son reportados en el artículo y los obtenidos como resultado estable en las simulaciones en los mismos puntos.

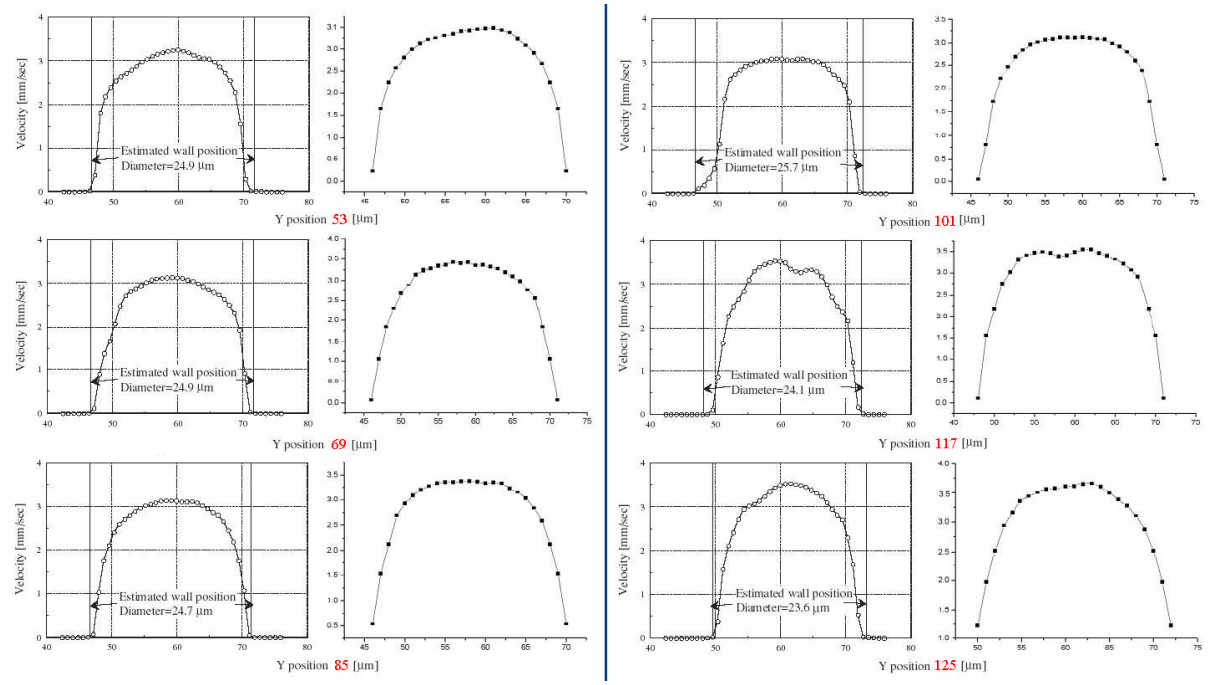


Figura 4.4. Perfiles de velocidad experimentales y los obtenidos en las simulaciones.

Estos perfiles de velocidad muestran que el comportamiento de la sangre es del tipo no Newtoniano. También se observan pequeñas diferencias entre las gráficas experimentales y las gráficas obtenidas mediante simulación en el modelo (ver figura 4.4). Estas pequeñas diferencias, según reporta el grupo de investigadores, es devido a que en la arteriola real existe adherencia de leucocitos y eritrocitos en las paredes de ésta. Hecho que fue confirmado usando observación directa vía microscópio [38] y además a que en el modelo no se recrearon todos los factores que afectan al flujo sanguíneo, como es el caso del factor de adherencia de plaquetas a las paredes de la arteriola.

El número establecido de iteraciones en el sistema para llegar a resultados estables fue de 100,000 ciclos máquina. Corriendo el sistema en una máquina PC Pentium 4 de 3.2 Ghz de velocidad con memoria de 512 Mbytes, el tiempo de procesamiento por cada simulación fue aproximadamente de 45 minutos.

4.4.- USO DEL MODELO DE ARTERIOLA COMO UN SISTEMA DE MEDICIÓN.

Después de décadas de investigación los científicos e investigadores están cada vez más cercas de crear sangre artificial. Esta de más mencionar la importancia que esto significa para el área médica ya que son múltiples los beneficios que trae para los pacientes y enfermos. No solo puede salvar la vida de personas que pierden sangre después de sufrir un accidente, sino ayuda también a erradicar enfermedades contagiosas que son transmisibles por vía sanguínea después de una transfusión como es el caso del sida o hepatitis.

Aunque se han desarrollado ya algunos prototipos de sangre artificial como es el caso del compuesto MP4, que ha sido probado con relativo éxito en animales de laboratorio [9], el problema de absorción eficiente de nutrientes a nivel capilar aún siguen siendo una meta por superar [43].

La síntesis y liberación de óxido nítrico (NO) por el endotelio vascular esta determinada por factores de diversa naturaleza como los relacionados con la neurotransmisión, mediadores químicos y estímulos físicos. La fricción o esfuerzo de corte (shear stress), que ejerce la sangre sobre el endotelio vascular, es un factor importante de naturaleza física que induce la síntesis de NO en este tejido [43] y puede ser la causa de diversas enfermedades cardiovasculares entre las que destacan la trombosis arterial y la aterosclerosis.

Por esta razón en este apartado se estudiará el comportamiento del esfuerzo de corte bajo condiciones reales de la sangre.

Intaglietta en 2003 logró medir el nivel de viscosidad de la sangre en una arteriola y publicó sus resultados en el artículo “Microvascular Pressure and Functional Capillary Density in Extreme Hemodilution with Low and High Viscosity Dextran and a Low Viscosity Hb-Based O_2 Carrier “ [43]. Intaglietta comenta que bajo condiciones normales el nivel de viscosidad de la sangre en una arteriola es del orden de 4.21 cP.

Este dato permite calcular por medio de un planteamiento de ley de la potencia de Ostwald, el índice de comportamiento ‘n’ y el índice de consistencia ‘k’ del fluido. Información importante ya que por medio de ella, del modelo de arteriola y de los perfiles de velocidad publicados en el artículo ‘Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique’ [56], se puede calcular el esfuerzo de corte y el nivel de presión que puede existir en una arteriola con ese grado de viscosidad de la sangre.

A). Cálculo de la velocidad de corte, esfuerzo de corte y cambios de presión en las paredes de la arteriola.

El esfuerzo de corte se define como: $\tau = \mu \frac{dv}{dy}$, y representa la fuerza por unidad de área para mover un fluido. Donde μ es la viscosidad dinámica para fluidos newtonianos o la viscosidad aparente para fluidos no newtonianos. Y dv/dy es la velocidad de corte que representa el cambio de la velocidad de las capas de un fluido en un fluido laminar.

Tenemos que la viscosidad aparente (μ) para un fluido no newtoniano es definida por la ley de la potencia de Ostwald:

$$\mu = k\left(\frac{dv}{dy}\right)^{n-1}$$

donde 'k' y 'n' son el índice de consistencia y el índice de comportamiento del fluido respectivamente. Por lo tanto para un fluido no newtoniano el esfuerzo de corte es reducido a:

$$\tau = k\left(\frac{dv}{dy}\right)^{n-1}\frac{dv}{dy} = k\left(\frac{dv}{dy}\right)^{n} \qquad (4.6)$$

- **Cálculo de la velocidad de corte en las paredes de la arteriola.**

La solución discreta para la velocidad de corte (dv/dy) mediante diferencias divididas finitas centradas es [47] :

$$\frac{dv}{dy} = \frac{-v_{i+2} + 8\,v_{i+1} - 8\,v_{i-1} + v_{i-2}}{12h} \qquad (4.7)$$

donde v_i denota el valor de 'v' en la posición ' i ' y 'h' es el tamaño de peso que establece el valor de separación entre un valor de muestra v_i y su valor sucesor v_{i+1} o su valor predecesor v_{i-1}. Para este caso h=1, debido a que la separación entre velocidades v_i en la rejilla discreta de Boltzmann es de 1 nodo.

El siguiente ejemplo ilustra como es calculado dimensionalmente el valor de la velocidad de corte para un determinado punto cercano a la pared de la arteriola. Los valores de las velocidades v_{i-2}, v_{i-1}, v_{i+1} y v_{i+2} son valores inicialmente adimensionales tomados del modelo en cuatro puntos contiguos a la pared. Ver figura 4.5.

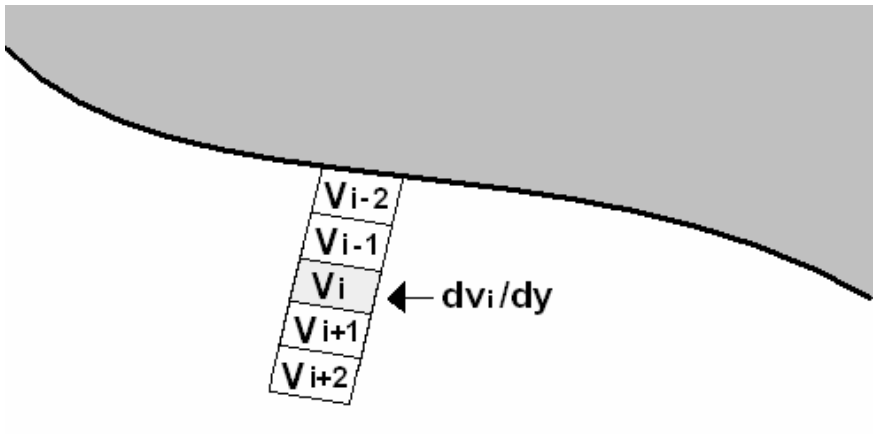


Figura 4.5. Puntos contíguos a la pared de la arteriola.

Los valores adimensionales de las velocidades obtenidos del modelo son : $v_{i-2}=0$, $v_{i-1}=1.59\text{x}10^{-7}$, $v_{i+1}=3.94\text{x}10^{-7}$ y $v_{i+2}=4.90\text{x}10^{-7}$, los cuales convertidos a valores dimensionales son: (ver apartado 4.3)

$v_{i-2}=0$ mts/seg,
$v_{i-1}=0.2\text{x}10^{-3}$ mts/seg,
$v_{i+1}=0.62\text{x}10^{-3}$ mts/seg y
$v_{i+2}=0.77\text{x}10^{-3}$ mts/seg

Convirtiendo a valor dimensional el valor de 'h=1' se obtiene que h=1x10^{-6} mts. Por lo que evaluando con estos datos la ecuación (4.7) se obtiene como resultado:

$$\frac{dv_i}{dy}=\frac{-0.77\text{x}10^{-3}+8\left(0.62\text{x}10^{-3}\right)-8\left(0.25\text{x}10^{-3}\right)+0}{12\left(1\text{x}10^{-6}\right)}=182.5\ \text{seg}^{-1}$$

Alimentando al modelo con los valores adimensionales de k=2.08x10^{-5} y n=0.708 manejados en el apartado 4.2 inciso (C) para calcular el parámetro de relajación 'τ ' en el sistema (ver apartado 1.10, ecuación 1.68) y con los valores adimensionales de las velocidades de entrada (2.7 mm/seg) y de salida (3.5 mm/seg) de una arteriola publicados en [56], se obtiene evaluando la ecuación (4.7) en el modelo de arteriola, los valores teóricos de velocidad de corte en los 136 puntos que conforman las paredes superior e inferior del segmento de arteriola. (ver figura 4.6)

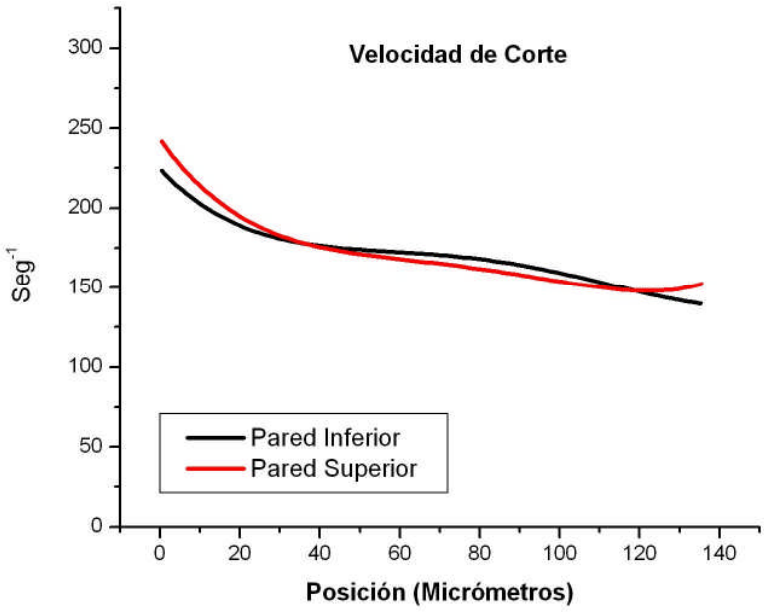


Figura 4.6. Comportamiento de la velocidad de corte en ambas paredes de la arteriola.

El valor de la velocidad de corte en la pared inferior tiene un valor medio de 168.31 seg^{-1} y en la pared superior de 169.71 seg^{-1}.

- **Cálculo del esfuerzo de corte en las paredes de la arteriola.**

El valor del esfuezo de corte en las paredes de la arteriola es calculado evaluando la ecuación (4.6) con los valores de la velocidad de corte obtenidos previamente y con base al valor dimensional de k=16.66x10^{-3} Pa·s^{n} y de n=0.708 publicados en [45].

Por ejemplo para el valor de la velocidad de corte calculado en el apartado anterior se tiene que el valor del esfuerzo de corte es:

$$\tau = k\left(\frac{dv}{dy}\right)^{n-1}\frac{dv}{dy} = k\left(\frac{dv}{dy}\right)^{n} = 16.66\text{x}10^{-3}\ \text{Pa}\cdot\text{seg}^{n}\left(182.5\ \text{seg}^{-1}\right)^{0.708} = 0.66\ \text{Pa}$$

La figura 4.7 muestra el valor teórico del esfuerzo de corte en los 136 puntos que conforman las paredes del segmento de arteriola.

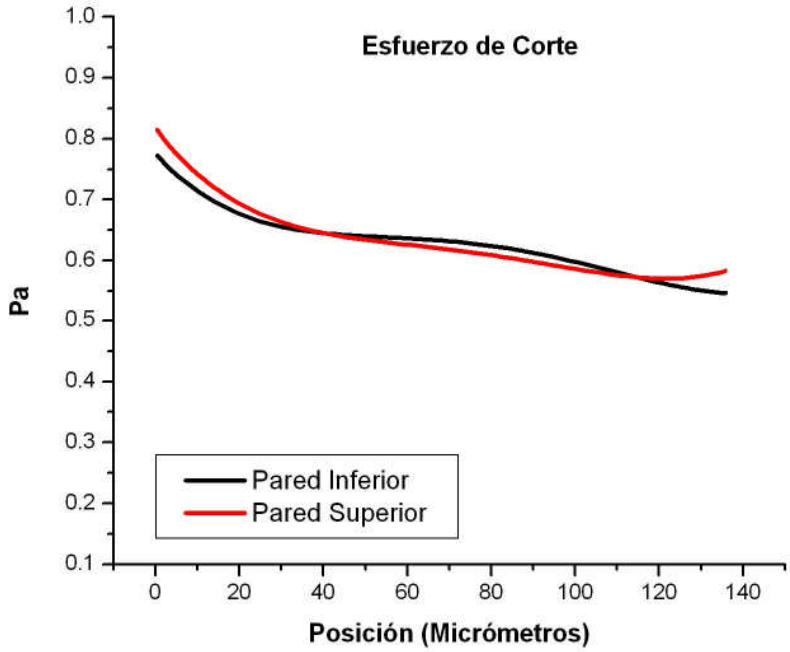


Figura 4.7. Comportamiento del esfuerzo de corte en ambas paredes de la arteriola.

El valor del esfuerzo de corte en la pared inferior tiene un valor medio de 0.628 Pa y en la pared superior de 0.649 Pa.

- **Cálculo de los valores de presión en las paredes de la arteriola.**

Alimentando al modelo con los valores de $k=2.08x10^{-5}$ y $n=0.708$ manejados en el apartado 4.2 inciso (C), se obtiene mediante la evaluación de la ecuación (4.5) en el modelo, los siguientes valores teóricos de presión dimensional expresados en Pascales en las paredes del segmento de arteriola. (ver figura 4.8)

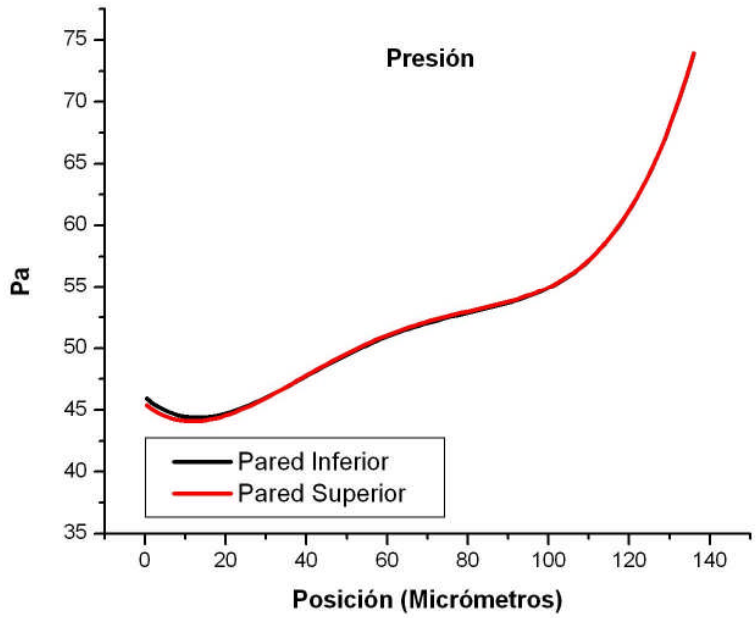


Figura 4.8. Comportamiento de la presión en ambas paredes de la arteriola.

El valor de la presión en la pared inferior tiene un valor medio de 52.59 Pa y en la pared superior de 52.58 Pa.

B). Comportamiento de la velocidad de corte, esfuerzo de corte y presión bajo diferentes niveles de viscosidad de la sangre humana.

Diferentes valores de viscosidad de la sangre publicados por Intaglietta [43] son mostrados en la tabla 4.6.

Fluido	Viscosidad de la sangre (cP)	Viscosidad de la sangre (Pa·s)
sangre	4.21	4.21×10^{-3}
Dextran 500	2.88	2.88×10^{-3}
Dextran 70	2.11	2.11×10^{-3}
PBH	1.72	1.72×10^{-3}

Tabla 4.6. Diferentes grados viscosidad para la sangre humana.

Donde el nivel de viscosidad de la sangre humana promedio es alterado con fines experimentales mediante diferentes químicos: Dextran 70, Dextran 500 y PBH.

Por otro lado se tiene que el valor de la viscosidad aparente (μ) para fluidos no Newtonianos de acuerdo a la ley de la potencia de Ostwald es:

$$\mu = k\left(\frac{dv}{dy}\right)^{n-1} \tag{4.8}$$

Donde ‘k’ y ‘n’ son el índice de consistencia e índice de comportamiento del fluido respectivamente.

Despejando a ‘k’ de la ecuación (4.8):

$$\mathrm{k} = \frac{\mu}{\left(\frac{dv}{dy}\right)^{\mathrm{n\text{-}1}}} \tag{4.9}$$

Donde (dv/dy) es el valor de la velocidad de corte (shear rate).

Para convertir el valor de ‘k’ dimensional a valor adimensional necesario para alimentar al modelo de arteriola, se toma en cuenta el valor dimensional de $k=16.66 \times 10^{-3}$ $Pa \cdot seg^n$ publicado en [45] y el valor adimensional de $k=2.08 \times 10^{-5}$ calculado en el apartado 4.2 inciso (C). Para ésto se calcula un **valor base** de ‘k’ **dimensional** (K_{BD}) el cual sirve como factor de adimensionalización.

Tomando el valor de k=16.66x10^{-3} Pa·seg^{n} como K_D y el valor de k=2.08x10^{-5} como K_A , mediante la siguiente expresión se calcula el valor de K_{BD}.

$$\frac{K_{BD}}{K_D} = K_A \,, \tag{4.10}$$

Despejando y evaluando a expresión (4.10) se obtiene:

$$K_{BD} = (K_A)(K_D) = \left(2.08x10^{-5}\right)\left(16.66x10^{-3}\ Pa \cdot seg^{n}\right) = 3.4652x10^{-7}\ Pa \cdot seg^{n}$$

Por lo que para obtener un valor adimensional de 'k' calculado mediante la ecuación (6.9) se evalua la ecuación (4.11) la cual toma como referencia el valor de K_{BD} calculado previamente y un valor de viscosidad de la tabla 6.6 expresado en Pa·seg y el valor promedio de la velocidad de corte calculado previamente en el apartado 4.4 inciso (A) expresado en seg^{-1}.

$$K_A = \frac{K_{BD}}{K_D} = \frac{3.4652x10^{-7}\ Pa \cdot seg^{n}}{\left[\dfrac{Viscosidad(\mu)}{\left(169\ seg^{-1}\right)^{n-1}}\right]} \tag{4.11}$$

La tabla 4.7 muestra algunos valores de 'k' adimensional y de 'n' para los diferentes valores de viscosidad (μ) de la tabla 4.6 que satisfacen la ecuación (4.11).

Viscosidad de la sangre Pa·s	**Indice 'n' adimensional**	**Indice 'k' adimensional**
4.21 x10^{-3}	0.9	4.92x10^{-5}
	0.7	1.76x10^{-5}
	0.5	6.33x10^{-6}
2.88 x10^{-3}	0.9	7.20x10^{-5}
	0.7	2.58x10^{-5}
	0.5	9.25x10^{-6}

Tabla 4.7 (A). Valores de 'k' y de 'n' que satisfacen la ecuación (6.11).

Viscosidad de la sangre Pa·s	Indice 'n' adimensional	Indice 'k' adimensional
2.11×10^{-3}	0.9	9.83×10^{-5}
	0.7	3.52×10^{-5}
	0.5	1.26×10^{-5}
1.72×10^{-3}	0.9	1.20×10^{-4}
	0.7	4.32×10^{-5}
	0.5	1.54×10^{-5}

Tabla 4.7 (B). Valores de 'k' y de 'n' que satisfacen la ecuación (6.11).

- **Comportamiento de la velocidad de corte en las paredes de la arteriola.**

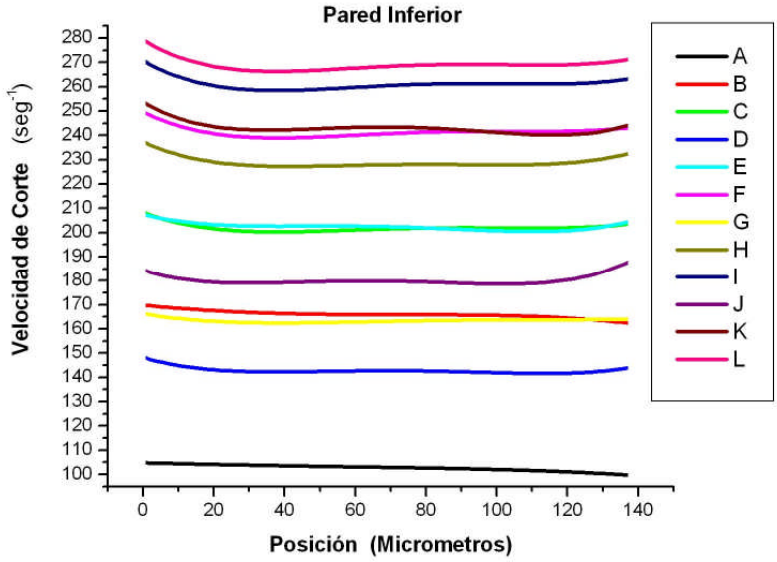


Figura 4.9. Comportamiento de la velocidad de corte en la pared infeior de la arteriola.

La tabla 4.8 muestra el valor promedio de la velocidad de corte en la pared inferior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Velocidad de Corte** Promedio (Seg^{-1})
A	4.21×10^{-3}	0.5	102.79
B		0.7	166.09
C		0.9	201.94
D	2.88×10^{-3}	0.5	142.78
E		0.7	202.43
F		0.9	241.32
G	2.11×10^{-3}	0.5	163.46
H		0.7	228.79
I		0.9	261.02
J	1.72×10^{-3}	0.5	180.12
K		0.7	242.95
L		0.9	268.93

Tabla 4.8. Valores promedio de la velocidad de corte para diferentes valores de viscosidad.

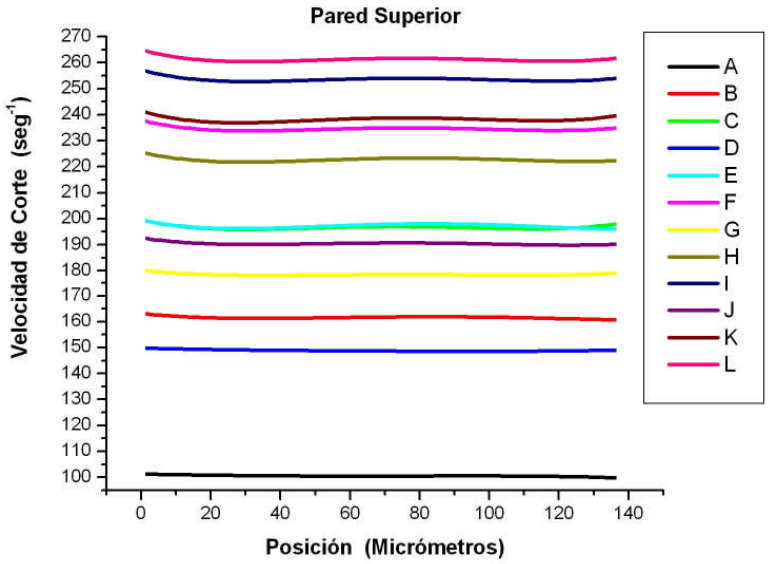


Figura 4.10. Comportamiento de la velocidad de corte en la pared superior de la arteriola.

La tabla 4.9 muestra el valor medio de la velocidad de corte en la pared superior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Velocidad de Corte** Promedio (Seg^{-1})
A	4.21×10^{-3}	0.5	100.56
B		0.7	161.62
C		0.9	196.69
D	2.88×10^{-3}	0.5	148.91
E		0.7	197.20
F		0.9	234.55
G	2.11×10^{-3}	0.5	178.58
H		0.7	222.71
I		0.9	253.61
J	1.72×10^{-3}	0.5	190.47
K		0.7	238.17
L		0.9	261.25

Tabla 4.9. Valores promedio de la velocidad de corte para diferentes valores de viscosidad.

- **Comportamiento del esfuerzo de corte en las paredes de la arteriola.**

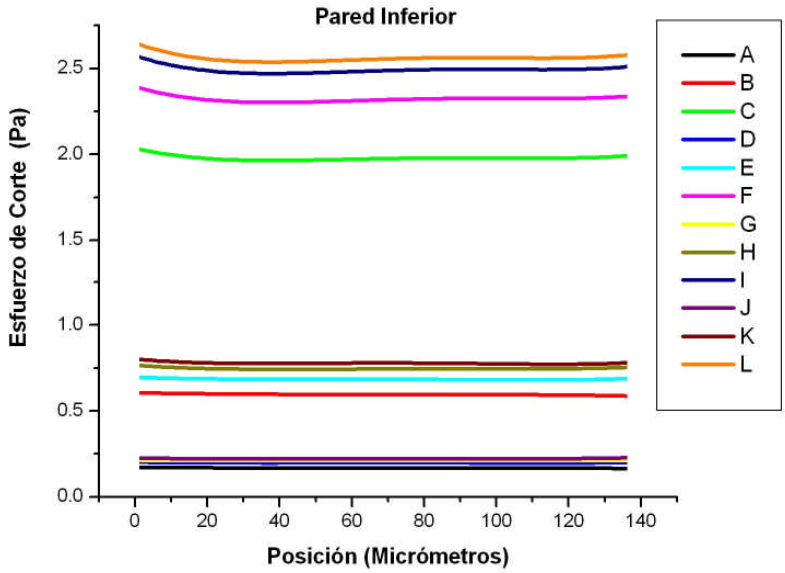


Figura 4.11. Comportamiento del esfuerzo de corte en la pared infeior de la arteriola.

La tabla 4.10 muestra el valor medio del esfuerzo de corte en la pared inferior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Esfuerzo de Corte** Promedio (Pa)
A	4.21×10^{-3}	0.5	0.1688
B		0.7	0.5968
C		0.9	1.9786
D	2.88×10^{-3}	0.5	0.1990
E		0.7	0.6855
F		0.9	2.3227
G	2.11×10^{-3}	0.5	0.2129
H		0.7	0.7468
I		0.9	2.4927
J	1.72×10^{-3}	0.5	0.2235
K		0.7	0.7789
L		0.9	2.5606

Tabla 4.10. Valores promedio del esfuerzo de corte para diferentes valores de viscosidad.

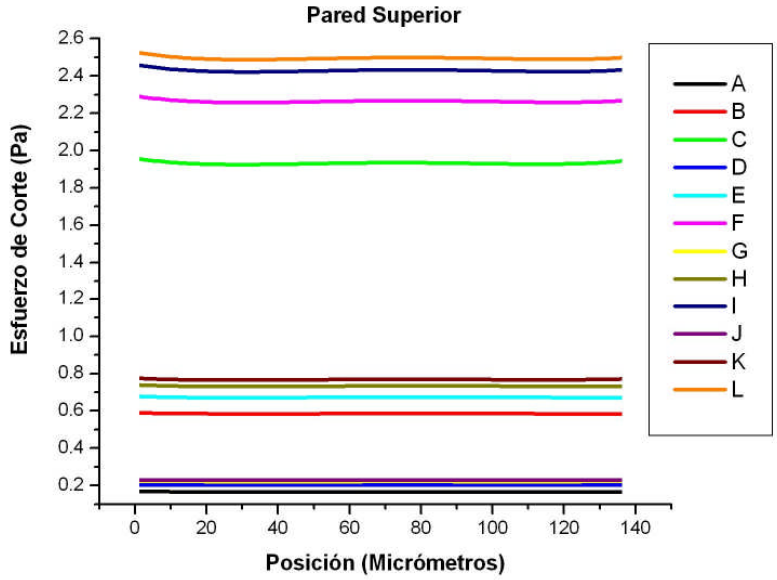


Figura 4.12. Comportamiento del esfuerzo de corte en la pared superior de la arteriola.

La tabla 4.11 muestra el valor medio del esfuerzo de corte en la pared superior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Esfuerzo de Corte** Promedio (Pa)
A	4.21×10^{-3}	0.5	0.1670
B		0.7	0.5856
C		0.9	1.9323
D	2.88×10^{-3}	0.5	0.2033
E		0.7	0.6731
F		0.9	2.2641
G	2.11×10^{-3}	0.5	0.2226
H		0.7	0.7329
I		0.9	2.4289
J	1.72×10^{-3}	0.5	0.2299
K		0.7	0.7682
L		0.9	2.4947

Tabla 4.11. Valores promedio del esfuerzo de corte para diferentes valores de viscosidad.

- **Comportamiento de la presión en las paredes de la arteriola.**

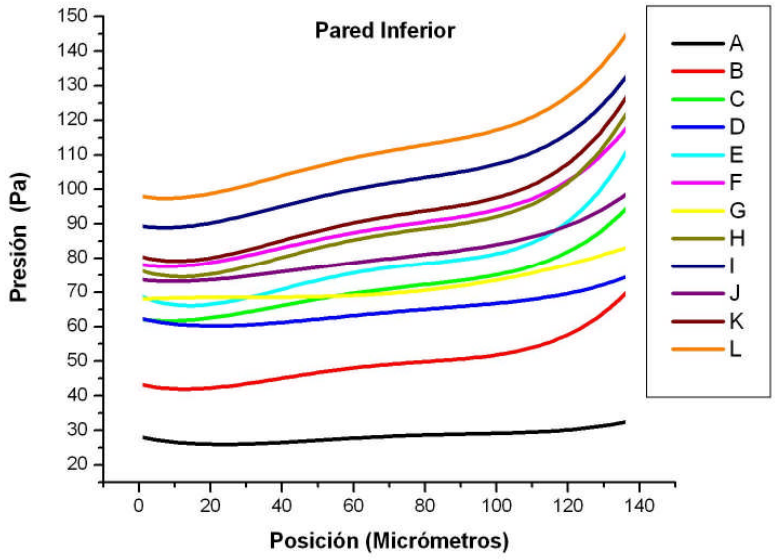


Figura 4.13. Comportamiento de la presión en la pared infeior de la arteriola.

La tabla 4.12 muestra el valor medio de la presión en la pared inferior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Presión** Promedio (Pa)
A	4.21×10^{-3}	0.5	28.21
B		0.7	49.64
C		0.9	71.67
D	2.88×10^{-3}	0.5	64.63
E		0.7	78.09
F		0.9	89.88
G	2.11×10^{-3}	0.5	71.74
H		0.7	88.02
I		0.9	102.54
J	1.72×10^{-3}	0.5	80.81
K		0.7	93.08
L		0.9	112.05

Tabla 4.12. Valores promedio de la presión para diferentes valores de viscosidad.

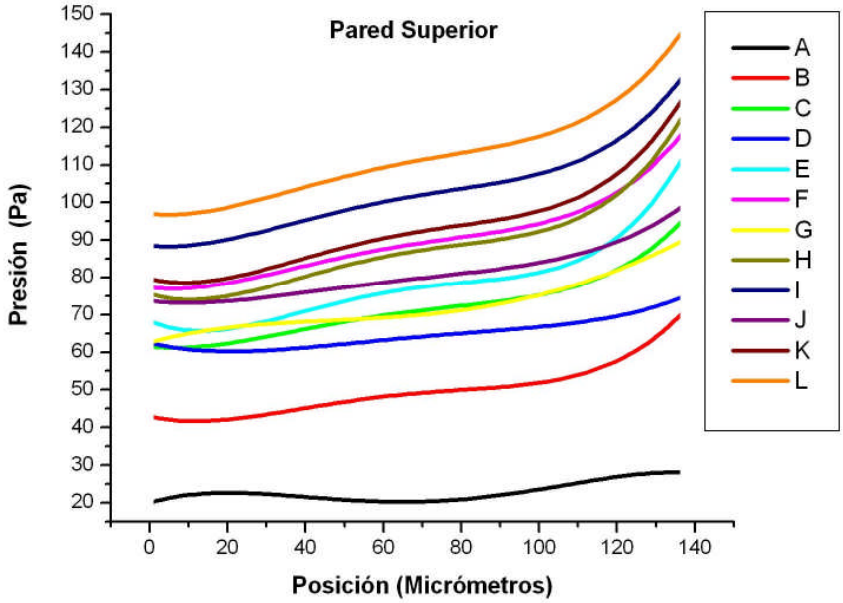


Figura 4.14. Comportamiento de la presión en la pared superior de la arteriola.

La tabla 4.13 muestra el valor medio de la presión en la pared superior de la arteriola correspondiente a cada nivel de viscosidad manejado.

Indice	**Viscosidad de la sangre** (Pa·s)	**n**	**Presión** Promedio (Pa)
A	4.21×10^{-3}	0.5	22.80
B		0.7	49.63
C		0.9	71.71
D	2.88×10^{-3}	0.5	64.63
E		0.7	78.08
F		0.9	89.93
G	2.11×10^{-3}	0.5	72.30
H		0.7	88.02
I		0.9	102.66
J	1.72×10^{-3}	0.5	80.81
K		0.7	93.07
L		0.9	112.17

Tabla 4.13. Valores promedio de la presión para diferentes valores de viscosidad.

DISCUSIÓN

Diversas investigaciones en el tema de la dinámica de fluidos muestran que existe un esfuerzo de pared cortante crítico que ayuda a limpiar de impurezas, tales como depósitos de bacterias u hongos, las paredes internas de los ductos por el cual circulan los fluidos [66]. Sin embargo, en la fisiología hemodinámica arterial no existe evidencia alguna que apoye éste concepto. Por el contrario, existe evidencia de que el incremento en la pared de las arterias del esfuerzo de corte, cuyo valor normal promedio varía entre 0.5 y 2.0 Pa [48], provoca que los leucocitos y eritrocitos tiendan a circular por la parte central del flujo y las plaquetas, también llamadas trombocitos, por la parte periférica del flujo [49,50]. Esto hace que exista una mayor adherencia de plaquetas en el endotelio que es la membrana que recubre las paredes interiores de los vasos sanguíneos, ya que el aumento y acumulación de plaquetas aumenta su cualidad adhesiva, lo que también favorece la formación de coágulos en los vasos.

Como la función de las plaquetas es evitar la pérdida de la sangre cohibiendo las hemorragias, la acumulación de plaquetas en el endotelio hace que en principio se de una aceleración del proceso regenerativo de las células dañadas. Aunque cabe destacar que si el aumento del esfuerzo corte permanece por un tiempo prolongado (superior a 2.0 Pa en arterias), la probabilidad de daño celular es alta. Esto debido a que las células pueden morir por inanición o falta de nutrientes, ya que los eritrocitos que son los encargados de llevar nutrientes a las células no pueden adherirse al endotelio para su asimilación, o por la obstrucción de los vasos sanguíneos debido a la formación de coágulos en la sangre. Por otro lado, si el nivel del esfuerzo de corte en las paredes de los vasos sanguíneos permanece relativamente bajo durante un tiempo prolongado (inferior a 0.5 Pa en arteriolas) la obstrucción de vasos también es posible debido a que el endotelio libera sustancias tales como el óxido nítrico que dilatan el diámetro de los vasos [48].

Los resultados obtenidos de las simulaciones muestran que el comportamiento del esfuerzo de corte y de presión en las paredes de una arteriola esta en función del nivel de viscosidad de la sangre. El decremento del nivel de viscosidad de la sangre favorece el incremento del valor del esfuerzo de corte y niveles de presión en las paredes de los vasos sanguíneos. Los datos indican que un decremento de la viscosidad del 31%, 49% y 59% se relaciona con un incremento del shear stress del 15%, 25% y 30% respectivamente. Por otro lado, los datos también indican que un decremento de la viscosidad del 31%, 49% y 59% producen un incremento de la presión del 57%, 77% y 87% respectivamente. Esto favorece como ya se comento, la adherencia de plaquetas en el endotelio y la hipertensión.

Las gráficas muestran que en una reducción del diámetro de una arteriola el esfuerzo de corte es mayor cuando el nivel de viscosidad es menor, mientras que en un aumento del diámetro arterial el esfuerzo de corte en las paredes es menor cuando el nivel de viscosidad es menor. Si se toma en cuenta el valor normal promedio del esfuerzo de corte en una arteriola, el cual es 0.5 Pa [48].

El análisis de los resultados induce a pensar que en diámetros arteriales relativamente más estrechos el grado de regeneración celular en principio es menor que en otras áreas de las arteriolas. Y que también estas zonas son más propensas a formar coágulos que pueden desencadenar en lesiones graves como trombosis y muerte celular.

En conclusión, del modelo de arteriola propuesto se puede decir que es un modelo predictivo útil para conocer con base a niveles dados de viscosidad de la sangre, valores de la velocidad de corte, esfuerzo de corte y de presión en los distintos puntos que integran las paredes de los vasos sanguíneos (endotelio). Conocimiento útil y necesario para detectar los niveles de viscosidad de la sangre que pueden conducir a un daño celular.

Conclusiones

Del análisis de este trabajo de investigación se pueden desprender tres conclusiones. (1) La primera sobre los mecanismos de fronteras abiertas analizados. (2) La segunda sobre el análisis de la eficiencia del método de la ecuación de Boltzmann en rejillas y (3) la tercera sobre posibles aplicaciones futuras del producto de esta tesis.

(1) Debido al reciente nacimiento del tema de rejillas de Boltzmann y la creciente cantidad y variedad de problemas reales en los que se esta aplicando, se impide que en este momento exista un método o técnica refinada en fronteras abiertas que solucione todos los posibles casos de dinámica de fluidos del mundo real.

En este trabajo se analizaron dos mecanismos de fronteras abiertas en rejillas de Boltzmann publicados en años recientes, en los que se encontraron ciertas deficiencias con respecto a la solución del problema central de esta tesis. Sin embargo, esto no les resta importancia, ya que son útiles en la solución de otro tipo de problemas en los que se desea conocer el comportamiento de fluidos en ductos con geometrías internas rectas. Además, estos dos mecanismos conforman la base teórica del funcionamiento de los mecanismos de fronteras abiertas, y le permiten al lector obtener el conocimiento necesario para la creación de nuevos y mejores mecanismos de este tipo.

(2) No es duda que las dimensiones de la rejilla es uno de los factores que afectan principalmente la eficacia de LBE, ya que por cada columna o renglón de nodos que se aumenta en la rejilla el tiempo de simulación aumenta en forma significativa. Para ilustrar esto podemos tomar como ejemplo un modelo de rejilla D2Q9 de 10 nodos de largo por 10 nodos de ancho. Suponiendo un tiempo de procesamiento de 1 millonésima de segundo para la actualización de cada población f_i en cada nodo. Para actualizar toda la rejilla se necesitaría 9x10x10=900 millonésimas de segundo. El agregar una columna más de nodos a la rejilla significaría agregar 90 millonésimas de segundos más al tiempo de procesamiento.

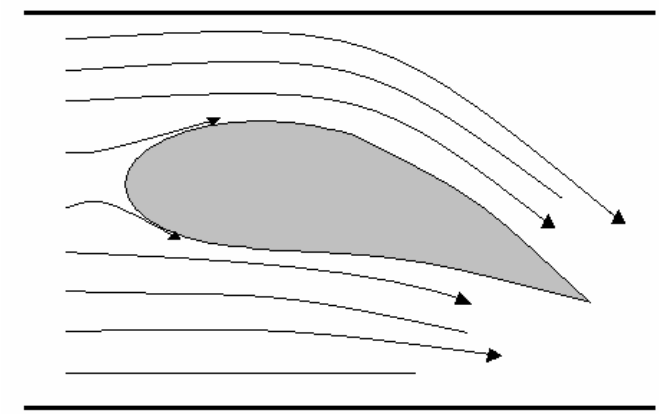
Si éste análisis lo llevamos al ámbito real del problema que se abordo en esta tesis. Para una rejilla de 136 nodos por 136 nodos con 9 poblaciones por actualizar por nodo, el tiempo por iteración para actualizar todas las poblaciones de la rejilla es de 9x136x136=166,464 millonésimas de segundo. Si esto lo multiplicamos por 150,000 iteraciones que se requieren para llegar a un estado estable en el que los resultados ya no presentan variaciones. El tiempo de computo requerido para una simulación es del orden de $166464x150000=2.49696x10^{10}$ millonésimas de segundo. Lo que significa que en tiempo cuantificable para un ser humano es de $2.49696x10^{4}$ segundos. Esto es 6.936 horas de cómputo por simulación. En un a microcomputadora con un procesador Pentium 4 a 3.2 GHz de velocidad, con 512 MBytes de memoria principal este tiempo se reduce a 45 minutos. Que como se podrá ver aún sigue siendo muy alto, si estamos hablando que es el tiempo requerido por cada simulación.

Si analizamos la secuencia de pasos mostrada en el apartado 3.9 nos daremos cuanta que existen áreas de oportunidad para mejorar desde un punto de vista computacional la eficiencia en tiempo del método de la ecuación de Boltzmann en rejillas. La propuesta para mejorar el tiempo consiste en paralelizar ciertos procesos del algoritmo. Esto es muy fácil si se traba con supercomputadoras, las cuales por naturaleza trabajan bajo este esquema. Pero si se trabaja en microcomputadoras, las cuales usan un solo microprocesador para realizar los cálculos, se puede obtener un rendimiento significativo si el concepto de paralelismo se lleva a cabo mediante hilos de ejecución.

La idea consiste en paralelizar la actualización de las poblaciones f_i en cada nodo de la rejilla, ya que son una vez calculadas las funciones de distribución de equilibrio f_i^{eq}, la actualización de las f_i es independiente una de la otra. (ver apartado 2.2, código del quinto proceso). Con esto se lograría reducir en un octavo el tiempo de cómputo requerido para cada simulación. Esto es de 1 hora con 50 minutos, se puede pasar a 5 minutos aproximadamente por simulación.

(3) Un mecanismo de fronteras abiertas en rejillas de Boltzmann para simular el comportamiento de fluidos en ductos abiertos puede tener múltiples aplicaciones y en diversas áreas de las ciencias puede ser útil. Entre algunas áreas probables destacan: la medicina, en el diseño y mejora de equipos de asistencia cardiaca o pulmonar, como puede ser el caso de válvulas cardiacas artificiales o respiradores artificiales. Otra área de aplicación puede ser la aeronáutica, en el análisis y diseño de naves de vuelo. Otra área puede ser la arquitectura, en el análisis de resistencia de estructuras a catástrofes naturales como pueden ser huracanes o tornados.

Por ejemplo en la asistencia del diseño de naves de vuelo mediante túneles de viento se puede tener el siguiente diseño de rejilla de Boltzmann con fronteras abiertas.



En donde el diseño del ala puede ser fácilmente manipulado para analizar las presiones que se pueden presentar por abajo y por arriba del ala y así establecer los requerimientos de fuerza automotriz que requiere el motor o turbina del avión.

Este mismo principio puede se aplicado en el diseño de las élises de un navío o submarino.

```
  1: unit Unit1;
  2:
  3: interface
  4:
  5: uses
  6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  7:   ExtCtrls, ComCtrls, Buttons, StdCtrls, Math, Gauges, ExtDlgs;
  8:
  9: type
 10:   TForm1 = class(TForm)
 11:     PageControl1: TPageControl;
 12:     TabSheet1: TTabSheet;
 13:     TabSheet2: TTabSheet;
 14:     Panel1: TPanel;
 15:     Image1: TImage;
 16:     Panel4: TPanel;
 17:     SpeedButton1: TSpeedButton;
 18:     Panel6: TPanel;
 19:     Panel7: TPanel;
 20:     Image2: TImage;
 21:     Panel11: TPanel;
 22:     Image3: TImage;
 23:     TabSheet3: TTabSheet;
 24:     TabSheet4: TTabSheet;
 25:     Label9: TLabel;
 26:     Panel3: TPanel;
 27:     Label2: TLabel;
 28:     Label3: TLabel;
 29:     SpeedButton4: TSpeedButton;
 30:     Label11: TLabel;
 31:     Edit2: TEdit;
 32:     Edit3: TEdit;
 33:     Memo1: TMemo;
 34:     GroupBox1: TGroupBox;
 35:     Label7: TLabel;
 36:     Label13: TLabel;
 37:     Edit9: TEdit;
 38:     Edit12: TEdit;
 39:     GroupBox2: TGroupBox;
 40:     Label16: TLabel;
 41:     Label17: TLabel;
 42:     Edit13: TEdit;
 43:     Edit14: TEdit;
 44:     GroupBox3: TGroupBox;
 45:     Edit1: TEdit;
 46:     Panel10: TPanel;
 47:     Panel13: TPanel;
 48:     Timer1: TTimer;
 49:     Panel20: TPanel;
 50:     PaintBox5: TPaintBox;
 51:     Timer2: TTimer;
 52:     Panel26: TPanel;
 53:     PaintBox8: TPaintBox;
 54:     Panel25: TPanel;
 55:     PaintBox7: TPaintBox;
 56:     Panel21: TPanel;
 57:     PaintBox9: TPaintBox;
 58:     Panel27: TPanel;
 59:     PaintBox10: TPaintBox;
 60:     Label21: TLabel;
 61:     Edit15: TEdit;
 62:     Label22: TLabel;
 63:     Edit17: TEdit;
 64:     Label23: TLabel;
 65:     Edit18: TEdit;
 66:     SpeedButton9: TSpeedButton;
 67:     Label24: TLabel;
 68:     Edit19: TEdit;
 69:     Label25: TLabel;
 70:     Label26: TLabel;
 71:     Label27: TLabel;
 72:     Label28: TLabel;
 73:     ProgressBar1: TProgressBar;
```

```
  74:     Edit20: TEdit;
  75:     Edit21: TEdit;
  76:     Label29: TLabel;
  77:     Label30: TLabel;
  78:     OpenPictureDialog1: TOpenPictureDialog;
  79:     SpeedButton11: TSpeedButton;
  80:     Timer3: TTimer;
  81:     SpeedButton12: TSpeedButton;
  82:     Timer4: TTimer;
  83:     Timer5: TTimer;
  84:     Panel28: TPanel;
  85:     Image4: TImage;
  86:     Panel30: TPanel;
  87:     Image6: TImage;
  88:     Panel31: TPanel;
  89:     Image7: TImage;
  90:     Panel32: TPanel;
  91:     Image8: TImage;
  92:     Panel33: TPanel;
  93:     Image9: TImage;
  94:     Panel29: TPanel;
  95:     Image5: TImage;
  96:     SpeedButton13: TSpeedButton;
  97:     SpeedButton14: TSpeedButton;
  98:     Panel34: TPanel;
  99:     Image12: TImage;
 100:     Panel35: TPanel;
 101:     Image10: TImage;
 102:     Panel38: TPanel;
 103:     Panel39: TPanel;
 104:     Panel40: TPanel;
 105:     Panel41: TPanel;
 106:     GroupBox6: TGroupBox;
 107:     Label14: TLabel;
 108:     Edit10: TEdit;
 109:     Label15: TLabel;
 110:     Edit11: TEdit;
 111:     GroupBox7: TGroupBox;
 112:     Label5: TLabel;
 113:     Edit5: TEdit;
 114:     Panel2: TPanel;
 115:     PaintBox1: TPaintBox;
 116:     StaticText1: TStaticText;
 117:     StaticText3: TStaticText;
 118:     StaticText2: TStaticText;
 119:     StaticText4: TStaticText;
 120:     StaticText5: TStaticText;
 121:     Panel15: TPanel;
 122:     Panel24: TPanel;
 123:     Panel14: TPanel;
 124:     SpeedButton2: TSpeedButton;
 125:     SpeedButton3: TSpeedButton;
 126:     SpeedButton5: TSpeedButton;
 127:     SpeedButton10: TSpeedButton;
 128:     Panel5: TPanel;
 129:     Gauge1: TGauge;
 130:     ScrollBar1: TScrollBar;
 131:     Edit8: TEdit;
 132:     Panel16: TPanel;
 133:     SpeedButton6: TSpeedButton;
 134:     Label6: TLabel;
 135:     Label8: TLabel;
 136:     Label10: TLabel;
 137:     Label12: TLabel;
 138:     Label18: TLabel;
 139:     Label19: TLabel;
 140:     Label20: TLabel;
 141:     Label31: TLabel;
 142:     Label32: TLabel;
 143:     SpeedButton7: TSpeedButton;
 144:     SpeedButton8: TSpeedButton;
 145:     Label33: TLabel;
 146:     Label34: TLabel;
```

```
147:     Label35: TLabel;
148:     Label36: TLabel;
149:     Label38: TLabel;
150:     Label39: TLabel;
151:     Label40: TLabel;
152:     Label41: TLabel;
153:     Label42: TLabel;
154:     Label43: TLabel;
155:     Label44: TLabel;
156:     Label45: TLabel;
157:     Label1: TLabel;
158:     Panel17: TPanel;
159:     PaintBox2: TPaintBox;
160:     Panel18: TPanel;
161:     PaintBox3: TPaintBox;
162:     Panel19: TPanel;
163:     PaintBox4: TPaintBox;
164:     ScrollBar2: TScrollBar;
165:     Edit6: TEdit;
166:     ScrollBar3: TScrollBar;
167:     Edit7: TEdit;
168:     Memo2: TMemo;
169:     Memo3: TMemo;
170:     ScrollBar5: TScrollBar;
171:     Edit16: TEdit;
172:     Panel22: TPanel;
173:     Memo4: TMemo;
174:     Panel23: TPanel;
175:     Memo5: TMemo;
176:     Edit22: TEdit;
177:     ScrollBar4: TScrollBar;
178:     Edit24: TEdit;
179:     ScrollBar6: TScrollBar;
180:     Edit25: TEdit;
181:     Panel36: TPanel;
182:     RadioButton1: TRadioButton;
183:     RadioButton2: TRadioButton;
184:     Panel37: TPanel;
185:     GroupBox4: TGroupBox;
186:     Label46: TLabel;
187:     Label47: TLabel;
188:     Edit26: TEdit;
189:     Edit27: TEdit;
190:     GroupBox5: TGroupBox;
191:     Label48: TLabel;
192:     Label49: TLabel;
193:     Edit28: TEdit;
194:     Edit29: TEdit;
195:     PaintBox6: TPaintBox;
196:     Label4: TLabel;
197:     Panel8: TPanel;
198:     Label37: TLabel;
199:     Panel9: TPanel;
200:     Label50: TLabel;
201:     Panel12: TPanel;
202:     Label51: TLabel;
203:     SpeedButton15: TSpeedButton;
204:     SpeedButton16: TSpeedButton;
205:     Panel42: TPanel;
206:     PaintBox11: TPaintBox;
207:     Label52: TLabel;
208:     Panel45: TPanel;
209:     Panel43: TPanel;
210:     ScrollBar7: TScrollBar;
211:     Edit4: TEdit;
212:     Timer6: TTimer;
213:     Panel44: TPanel;
214:     Panel46: TPanel;
215:     Edit23: TEdit;
216:     Panel47: TPanel;
217:     SpeedButton17: TSpeedButton;
218:     Label53: TLabel;
219:     Label54: TLabel;
```

```
220:     Label55: TLabel;
221:     Label56: TLabel;
222:     procedure SpeedButton1Click(Sender: TObject);
223:     procedure PaintBox1Paint(Sender: TObject);
224:     procedure cuadricula;
225:     procedure FormCreate(Sender: TObject);
226:     procedure FormClose(Sender: TObject; var Action: TCloseAction);
227:     procedure SpeedButton4Click(Sender: TObject);
228:     procedure SpeedButton2Click(Sender: TObject);
229:     procedure ScrollBar1Change(Sender: TObject);
230:     procedure SpeedButton3Click(Sender: TObject);
231:     procedure SpeedButton5Click(Sender: TObject);
232:     procedure ScrollBar2Change(Sender: TObject);
233:     procedure grafica1;
234:     procedure grafica2;
235:     procedure PaintBox3Paint(Sender: TObject);
236:     procedure ScrollBar3Change(Sender: TObject);
237:     procedure SpeedButton6Click(Sender: TObject);
238:     procedure ScrollBar5Change(Sender: TObject);
239:     procedure PaintBox4Paint(Sender: TObject);
240:     procedure PaintBox2Paint(Sender: TObject);
241:     procedure PaintBox5Paint(Sender: TObject);
242:     procedure Timer1Timer(Sender: TObject);
243:     procedure Timer2Timer(Sender: TObject);
244:     procedure PaintBox7Paint(Sender: TObject);
245:     procedure SpeedButton9Click(Sender: TObject);
246:     procedure PaintBox10Paint(Sender: TObject);
247:     procedure PaintBox9Paint(Sender: TObject);
248:     procedure iniventanas;
249:     procedure SpeedButton10Click(Sender: TObject);
250:     procedure SpeedButton11Click(Sender: TObject);
251:     procedure Timer3Timer(Sender: TObject);
252:     procedure SpeedButton12Click(Sender: TObject);
253:     procedure VelEntrada;
254:     procedure VelSalida;
255:     procedure Timer4Timer(Sender: TObject);
256:     procedure Timer5Timer(Sender: TObject);
257:     procedure calculaMatrizDeltas;
258:     procedure SpeedButton14Click(Sender: TObject);
259:     procedure SpeedButton13Click(Sender: TObject);
260:     procedure ScrollBar4Change(Sender: TObject);
261:     procedure ScrollBar6Change(Sender: TObject);
262:     procedure SpeedButton15Click(Sender: TObject);
263:     procedure GraficaPlanaEntrada;
264:     procedure GraficaPlanaSalida;
265:     procedure ActivaVentanas;
266:     procedure Edit22Change(Sender: TObject);
267:     procedure Edit11Change(Sender: TObject);
268:     procedure InactivaVentanas;
269:     procedure PaintBox11Paint(Sender: TObject);
270:     procedure ScrollBar7Change(Sender: TObject);
271:     procedure PuntodeCorte(posicion:integer);
272:     procedure FormShow(Sender: TObject);
273:     procedure Timer6Timer(Sender: TObject);
274:     procedure PageControl1Change(Sender: TObject);
275:     procedure SpeedButton17Click(Sender: TObject);
276:   private
277:     { Private declarations }
278:   public
279:     { Public declarations }
280:   end;
281:
282: var
283:   Form1: TForm1;
284:
285: implementation
286:
287: uses Unit2;
288: {$R *.DFM}
289: const lx=136;
290:       ly=136;
291:
292: Type tipoarray1=array[0..8,1..lx,1..ly]of real;
```

```
293:      tipoarray2=array[1..lx,1..ly]of boolean;
294:      tipoarray3=array[0..8,1..ly]of real;
295:      tipomat=array[1..lx,1..ly] of real;
296:      rec=record
297:          x,y:real;
298:      end;
299:      tipomat2=array[1..lx,1..ly] of rec;
300:      rec2=record
301:          pos,vel:real;
302:      end;
303:
304: var node,n_hlp:tipoarray1;
305:     obst,obstBack:tipoarray2;
306:     n_hlpAux1,n_hlpAux2,DifVect:tipoarray3;
307:     density,accel,omega,ain,nin:real;
308:     t_max,cont,cont2:integer;
309:     {f1,f2:TextFile;
310:     buffer:string;
311:     fm1,fm2,fm3,fm4,}fm5:file;
312:     {Apfm1,Apfm2,Apfm3,Apfm4,}Apfm5:integer;
313:     UxE,UyE,UxS,UyS,DeltaUx,DeltaUy,UxIni,UyIni:real;
314:     bmp,bmpArteriola,bmp2,bmp3,bmp4,bmp5:tbitmap;
315:     FactorAjuste:real;
316:     Puntos:array[1..ly]of rec2;
317:     puntos2:array[1..lx]of rec2;
318:     Puntos3,puntos4:array[1..ly]of rec;
319:     Abortar:boolean;
320:     ValorAnterior,Diferencia:real;
321:     bmpgraf1,bmpgraf2,bmpgraf3,bmpgraf4:tbitmap;
322:     pini,pfin:integer;
323:     max,min:real;
324:     PerfilVelEnt,PerfilVelSal:array[1..150]of real;
325:     MatrizDeltas:tipomat2;
326:     Movio1,Movio2,Movio3,Movio4:boolean;
327:     posxcorte:integer;
328:
329: procedure Check_density(time:integer);
330: var x,y,n:integer;
331:     n_sum:real;
332:     st:string;
333: begin
334:     n_sum:=0;
335:     for y:=1 to ly do
336:         for x:=1 to lx do
337:             for n:=0 to 8 do
338:                 n_sum:=n_sum+node[n,x,y];
339:     st:=copy(floattostr(n_sum),pos(floattostr(n_sum),'.'),6);
340:     form1.Memo1.Lines.Add('T='+inttostr(time)+'    Masa='+st);
341: end;
342:
343: procedure Propagate;
344: var x,y,x_e,x_w,y_n,y_s,k:integer;
345: begin
346:      // Propagar valores
347:      for x:=1 to lx do
348:          for y:=1 to ly do begin
349:              y_n:=(y mod ly) +1;
350:              x_e:=(x mod lx) +1;
351:              y_s:=ly-((ly+1-y) mod ly);
352:              x_w:=lx-((lx+1-x) mod lx);
353:
354:              n_hlp[0,x,y]:=node[0,x,y];
355:              n_hlp[1,x_e,y]:=node[1,x,y];
356:              n_hlp[2,x,y_n]:=node[2,x,y];
357:              n_hlp[3,x_w,y]:=node[3,x,y];
358:              n_hlp[4,x,y_s]:=node[4,x,y];
359:              n_hlp[5,x_e,y_n]:=node[5,x,y];
360:              n_hlp[6,x_w,y_n]:=node[6,x,y];
361:              n_hlp[7,x_w,y_s]:=node[7,x,y];
362:              n_hlp[8,x_e,y_s]:=node[8,x,y];
363:
364:              if x_w=lx then begin  // <---- x=1
365:                 n_hlp[1,x,y]:=node[1,x,y];
```

```
366:                 n_hlp[5,x,y_n]:=node[5,x,y];
367:                 n_hlp[8,x,y_s]:=node[8,x,y];
368:             end;
369:             if x_e=1 then begin   // ----> x=lx
370:                 //***
371:             end;
372:         end;
373:
374:         for y:=1 to ly do
375:             for k:=0 to 8 do n_hlpAux1[k,y]:=n_hlp[k,lx,y];
376:
377:         for y:=1 to ly do begin
378:             y_n:=(y mod ly) +1;
379:             y_s:=ly-((ly+1-y) mod ly);
380:             n_hlp[3,lx,y]:=node[3,lx,y];
381:             n_hlp[6,lx,y_n]:=node[6,lx,y];
382:             n_hlp[7,lx,y_s]:=node[7,lx,y];
383:         end;
384:
385:         for y:=1 to ly do
386:             for k:=0 to 8 do n_hlpAux2[k,y]:=n_hlp[k,lx,y];
387:
388:         for y:=1 to ly do
389:             for k:=0 to 8 do DifVect[k,y]:=n_hlpAux1[k,y]-n_hlpAux2[k,y];
390:
391:         for y:=1 to ly do begin
392:             n_hlp[0,1,y]:=n_hlp[0,1,y]+DifVect[0,y];
393:             n_hlp[1,1,y]:=n_hlp[1,1,y]+DifVect[3,y];
394:             n_hlp[2,1,y]:=n_hlp[2,1,y]+DifVect[2,y];
395:             n_hlp[3,1,y]:=n_hlp[3,1,y]+DifVect[1,y];
396:             n_hlp[4,1,y]:=n_hlp[4,1,y]+DifVect[4,y];
397:             n_hlp[5,1,y]:=n_hlp[5,1,y]+DifVect[7,y];
398:             n_hlp[6,1,y]:=n_hlp[6,1,y]+DifVect[8,y];
399:             n_hlp[7,1,y]:=n_hlp[7,1,y]+DifVect[5,y];
400:             n_hlp[8,1,y]:=n_hlp[8,1,y]+DifVect[6,y];
401:         end;
402: end;
403:
404: procedure bounceback;
405: var x,y:integer;
406: begin
407:      for x:=1 to lx do
408:         for y:=1 to ly do
409:             if obst[x,y] then begin
410:                 node[1,x,y]:=n_hlp[3,x,y];
411:                 node[2,x,y]:=n_hlp[4,x,y];
412:                 node[3,x,y]:=n_hlp[1,x,y];
413:                 node[4,x,y]:=n_hlp[2,x,y];
414:                 node[5,x,y]:=n_hlp[7,x,y];
415:                 node[6,x,y]:=n_hlp[8,x,y];
416:                 node[7,x,y]:=n_hlp[5,x,y];
417:                 node[8,x,y]:=n_hlp[6,x,y];
418:             end;
419: end;
420:
421: procedure tform1.calculaMatrizDeltas;
422: var x,y,Apuntador,cont,k:integer;
423:     factor,pos:real;
424:     UxIni,DeltaUx:array[1..ly]of real;
425:     MatrizAuxDeltas,Maux:tipomat2;
426: begin
427:     //Fluido Newtoniano    FA= 0.0055
428:     //Fluido NO Newtoniano FA= 0.008
429:     for y:=1 to ly do DeltaUx[y]:=(PerfilVelEnt[y]-PerfilVelSal[y])/lx;
430:     for y:=1 to ly do UxIni[y]:=PerfilVelEnt[y]+DeltaUx[y];
431:
432:     //DeltaUx:=(UxE-UxS)/lx;
433:     //UxIni:=UxE+DeltaUx;
434:     for x:=1 to lx do begin
435:         //UxIni:=UxIni-DeltaUx;
436:         DeltaUy:=(UyE-UyS)/ly;
437:         UyIni:=UyE+DeltaUy;
438:         for y:=1 to ly do begin
```

```
439:             UxIni[y]:=UxIni[y]-DeltaUx[y];
440:             UyIni:=UyIni-DeltaUy;
441:             MatrizAuxDeltas[x,y].x:=UxIni[y]*FactorAjuste;
442:             MatrizAuxDeltas[x,y].y:=UyIni*Factor;
443:         end;
444:     end;
445:
446:     //Calculo de las deltas adaptadas al ducto tanto en X como en Y.
447:     for x:=1 to lx do
448:         for y:=1 to ly do begin
449:             MatrizDeltas[x,y].x:=0;
450:             MatrizDeltas[x,y].y:=0;
451:         end;
452:
453:     {for x:=1 to lx do begin
454:         cont:=0;
455:         for y:=1 to ly do if (not obst[x,y]) then cont:=cont+1;
456:         Pos:=ly/cont;
457:         Apuntador:=1;
458:         for y:=1 to ly do
459:             if (not obst[x,y]) then begin
460:                 MatrizDeltas[x,y].x:=MatrizAuxDeltas[x,round(Apuntador*pos)].x;
461:                 MatrizDeltas[x,y].y:=MatrizAuxDeltas[x,round(Apuntador*pos)].y;
462:                 Apuntador:=Apuntador+1;
463:             end;
464:     end;}
465:
466:     for x:=1 to lx do begin
467:         cont:=0;
468:         for y:=1 to ly do if (not obst[x,y]) then cont:=cont+1;
469:         Pos:=ly/cont;
470:         Apuntador:=0;
471:         for y:=1 to ly do
472:             if (not obst[x,y]) then begin
473:                 Apuntador:=Apuntador+1;
474:                 Maux[x,Apuntador].x:=MatrizAuxDeltas[x,round(Apuntador*pos)].x;
475:                 Maux[x,Apuntador].y:=MatrizAuxDeltas[x,round(Apuntador*pos)].y;
476:             end;
477:
478:         for k:=1 to (apuntador div 2) do begin
479:             Maux[x,apuntador-k+1].x:=Maux[x,k].x;
480:             Maux[x,apuntador-k+1].y:=Maux[x,k].y;
481:         end;
482:
483:         apuntador:=0;
484:         for y:=1 to ly do
485:             if (not obst[x,y]) then begin
486:                 apuntador:=apuntador+1;
487:                 MatrizDeltas[x,y].x:=Maux[x,apuntador].x;
488:                 MatrizDeltas[x,y].y:=Maux[x,apuntador].y;
489:             end;
490:     end;
491:
492:     {x:=100; cont:=0;
493:     Memo1.Clear;
494:     for y:=1 to ly do
495:         if (not obst[x,y]) then begin
496:             cont:=cont+1;
497:             Memo1.Lines.Add(inttostr(cont)+'  '+floattostr(MatrizDeltas[x,y].x));
498:         end;}
499: end;
500:
501: procedure relaxation(density,omega:real;time:integer);
502: var x,y,i,contacumu_x:integer;
503:     c_squ,t_0,t_1,t_2,u_x,u_y,u_squ,d_loc,
504:     tem1,tem2,acumu_x,AcumTem1,AcumTem2,AcumTem3:real;
505:     u_n:array[1..8]of real;
506:     n_equ:array[0..8]of real;
507:     d_loc_aux,omega2:real;
508:     u_xant,u_xdes,u_xant1,u_xant2,u_xdes1,u_xdes2:real;
509: begin
510:     t_0:=4/9;
511:     t_1:=1/9;
```

```
512:     t_2:=1/36;
513:     c_squ:=1/3;
514:     for x:=1 to lx do begin
515:         for y:=1 to ly do begin
516:             if (not obst[x,y]) then begin
517:                 d_loc:=0;
518:                 for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y];
519:
520:                 u_x:=(n_hlp[1,x,y]+n_hlp[5,x,y]+n_hlp[8,x,y]
521:                        -(n_hlp[3,x,y]+n_hlp[6,x,y]+n_hlp[7,x,y]))/d_loc+MatrizDeltas[x,y].x;
522:
523:                 u_y:=(n_hlp[2,x,y]+n_hlp[5,x,y]+n_hlp[6,x,y]
524:                        -(n_hlp[4,x,y]+n_hlp[7,x,y]+n_hlp[8,x,y]))/d_loc+MatrizDeltas[x,y].y;
525:
526:                 u_squ:=u_x*u_x+u_y*u_y;
527:
528:                 u_n[1]:=u_x;
529:                 u_n[2]:=u_y;
530:                 u_n[3]:=-u_x;
531:                 u_n[4]:=-u_y;
532:                 u_n[5]:=u_x+u_y;
533:                 u_n[6]:=-u_x+u_y;
534:                 u_n[7]:=-u_x-u_y;
535:                 u_n[8]:=u_x-u_y;
536:
537:                 n_equ[0]:=t_0*d_loc*(1-u_squ/(2*c_squ));
538:
539:                 n_equ[1]:=t_1*d_loc*(1+u_n[1]/c_squ
540:                                      +power(u_n[1],2)/(2*power(c_squ,2))
541:                                      -u_squ/(2*c_squ));
542:                 n_equ[2]:=t_1*d_loc*(1+u_n[2]/c_squ
543:                                      +power(u_n[2],2)/(2*power(c_squ,2))
544:                                      -u_squ/(2*c_squ));
545:                 n_equ[3]:=t_1*d_loc*(1+u_n[3]/c_squ
546:                                      +power(u_n[3],2)/(2*power(c_squ,2))
547:                                      -u_squ/(2*c_squ));
548:                 n_equ[4]:=t_1*d_loc*(1+u_n[4]/c_squ
549:                                      +power(u_n[4],2)/(2*power(c_squ,2))
550:                                      -u_squ/(2*c_squ));
551:
552:                 n_equ[5]:=t_2*d_loc*(1+u_n[5]/c_squ
553:                                      +power(u_n[5],2)/(2*power(c_squ,2))
554:                                      -u_squ/(2*c_squ));
555:                 n_equ[6]:=t_2*d_loc*(1+u_n[6]/c_squ
556:                                      +power(u_n[6],2)/(2*power(c_squ,2))
557:                                      -u_squ/(2*c_squ));
558:                 n_equ[7]:=t_2*d_loc*(1+u_n[7]/c_squ
559:                                      +power(u_n[7],2)/(2*power(c_squ,2))
560:                                      -u_squ/(2*c_squ));
561:                 n_equ[8]:=t_2*d_loc*(1+u_n[8]/c_squ
562:                                      +power(u_n[8],2)/(2*power(c_squ,2))
563:                                      -u_squ/(2*c_squ));
564:
565:                 if form1.RadioButton1.checked then begin
566:                        for i:=0 to 8 do
567:                            node[i,x,y]:=n_hlp[i,x,y]+omega*(n_equ[i]-n_hlp[i,x,y]);
568:                 end;
569:
570:                 if form1.RadioButton2.checked then begin   // OK
571:                        if y>1 then begin
572:                           d_loc:=0;
573:                           for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-1];
574:                           u_xant:=(n_hlp[1,x,y-1]+n_hlp[5,x,y-1]+n_hlp[8,x,y-1]
575:                                  -(n_hlp[3,x,y-1]+n_hlp[6,x,y-1]+n_hlp[7,x,y-1]))/d_loc;
576:                        end
577:                        else u_xant:=u_x;
578:                        if y>2 then begin
579:                           d_loc:=0;
580:                           for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-2];
581:                           u_xant2:=(n_hlp[1,x,y-2]+n_hlp[5,x,y-2]+n_hlp[8,x,y-2]
582:                                   -(n_hlp[3,x,y-2]+n_hlp[6,x,y-2]+n_hlp[7,x,y-2]))/d_loc;
583:                        end
584:                        else u_xant2:=u_x;
```

```
585:                         if y<ly then begin
586:                            d_loc:=0;
587:                            for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+1];
588:                            u_xdes:=(n_hlp[1,x,y+1]+n_hlp[5,x,y+1]+n_hlp[8,x,y+1]
589:                                   -(n_hlp[3,x,y+1]+n_hlp[6,x,y+1]+n_hlp[7,x,y+1]))/d_loc;
590:                         end
591:                         else u_xdes:=u_x;
592:                         if y<(ly-1) then begin
593:                            d_loc:=0;
594:                            for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+2];
595:                            u_xdes2:=(n_hlp[1,x,y+2]+n_hlp[5,x,y+2]+n_hlp[8,x,y+2]
596:                                    -(n_hlp[3,x,y+2]+n_hlp[6,x,y+2]+n_hlp[7,x,y+2]))/d_loc;
597:                         end
598:                         else u_xdes2:=u_x;
599:                         if abs((-u_xdes2+8*u_xdes-8*u_xant+u_xant2)/12)>0 then begin
600:
     omega2:=ain*power(abs((-u_xdes2+8*u_xdes-8*u_xant+u_xant2)/12),nin-1);
601:                              omega2:=1/(3*omega2+0.5);
602:                         end
603:                         else omega2:=u_x;
604:                         for i:=0 to 8 do
605:                             node[i,x,y]:=n_hlp[i,x,y]+omega2*(n_equ[i]-n_hlp[i,x,y]);
606:                 end;
607:
608:             end;
609:         end;
610:     end;
611: end;
612:
613: procedure GuardaVectores;
614: var x,y,k,w,cont:integer;
615:     M:tipoMat2;
616:     acum,mayor,menor,d_loc,u_x,u_y,pos1,pos2:real;
617:     Deltas:array[1..lx]of real;
618: begin
619:     for x:=1 to lx do begin
620:         acum:=0; cont:=0;
621:         for y:=1 to ly do
622:             if not(obst[x,y]) then begin
623:                acum:=acum+y;
624:                cont:=cont+1;
625:             end;
626:         pos1:=acum/cont;
627:         acum:=0; cont:=0;
628:         for y:=1 to ly do
629:             if not(obstBack[x,y]) then begin
630:                acum:=acum+y;
631:                cont:=cont+1;
632:             end;
633:         pos2:=acum/cont;
634:         Deltas[x]:=pos2-pos1;
635:     end;
636:
637:     for x:=1 to lx do
638:         for y:=1 to ly do begin
639:             M[x,y].x:=0;
640:             M[x,y].y:=0;
641:         end;
642:     for x:=1 to lx do
643:         for y:=1 to ly do
644:             if (not obst[x,y]) then begin
645:                 d_loc:=0;
646:                 for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
647:
648:                 u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
649:                     -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
650:
651:                 u_y:=(node[2,x,y]+node[5,x,y]+node[6,x,y]
652:                     -(node[4,x,y]+node[7,x,y]+node[8,x,y]))/d_loc;
653:
654:                 M[x,y+round(Deltas[x])].x:=u_x;
655:                 M[x,y+round(Deltas[x])].y:=u_y;
656:             end;
```

```
657:     blockwrite(fm5,M,SizeOf(M));
658: end;
659:
660: procedure GuardaAnalisis1;
661: type rec=record
662:         vel:real;
663:         y:integer;
664:       end;
665: var  FV,FV2:TextFile;
666:       buffer:string;
667:       x,y,k,cont:integer;
668:       d_loc,u_x,u_y,pos1,pos2,acum:real;
669:       Deltas:array[1..lx]of real;
670:       Datos:array[1..ly]of rec;
671: begin
672:       for x:=1 to lx do begin
673:          acum:=0; cont:=0;
674:          for y:=1 to ly do
675:              if not(obst[x,y]) then begin
676:                 acum:=acum+y;
677:                 cont:=cont+1;
678:              end;
679:          pos1:=acum/cont;
680:          acum:=0; cont:=0;
681:          for y:=1 to ly do
682:              if not(obstBack[x,y]) then begin
683:                 acum:=acum+y;
684:                 cont:=cont+1;
685:              end;
686:          pos2:=acum/cont;
687:          Deltas[x]:=pos2-pos1;
688:       end;
689:
690:       assignfile(FV,'c:\Analisis1.txt');
691:       rewrite(FV);
692:       buffer:='Y              VelX';
693:       writeln(FV,buffer);
694:       x:=posxcorte;//101;
695:       cont:=0;
696:       for y:=ly downto 1 do
697:           if (not obst[x,y]) then begin
698:                 d_loc:=0;
699:                 for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
700:                 u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
701:                     -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
702:                 cont:=cont+1;
703:                 Datos[cont].vel:=u_x;
704:                 Datos[cont].y:=y;
705:           end;
706:       if (Datos[cont].vel<Datos[1].vel) then begin
707:           Datos[1].vel:=Datos[cont-0].vel;
708:           Datos[2].vel:=Datos[cont-1].vel;
709:           Datos[3].vel:=Datos[cont-2].vel;
710:       end;
711:       if (Datos[1].vel<Datos[cont].vel) then begin
712:           Datos[cont-0].vel:=Datos[1].vel;
713:           Datos[cont-1].vel:=Datos[2].vel;
714:           Datos[cont-2].vel:=Datos[3].vel;
715:       end;
716:
717:       assignfile(FV2,'c:\Tempo.txt');
718:       rewrite(FV2);
719:       for k:=1 to cont do begin
720:           buffer:=inttostr(ly-(Datos[k].y+round(Deltas[x])))+'       '+floattostr(Datos[k].vel);
721:           writeln(FV,buffer);
722:           writeln(FV2,floattostr(Datos[k].vel));
723:       end;
724:       closefile(FV);
725:       closefile(FV2);
726: end;
727:
728: procedure GraficaAnalisis1;
729: var  FV:TextFile;
```

```
730:      buffer:string;
731:      k,totalpuntos:integer;
732:      aux,maximo,minimo:real;
733: begin
734:       if Not FileExists('c:\Tempo.txt') then Exit;
735:       assignfile(FV,'c:\Tempo.txt');
736:       reset(FV);
737:       k:=0;
738:       while not(EOF(FV)) do begin
739:           k:=k+1;
740:           Puntos[k].pos:=k;
741:           readln(FV,Puntos[k].vel);
742:       end;
743:       closefile(FV);
744:       totalpuntos:=k;
745:
746:       maximo:=Puntos[1].vel;
747:       for k:=1 to totalpuntos do
748:           if Puntos[k].vel>maximo then maximo:=Puntos[k].vel;
749:       minimo:=Puntos[1].vel;
750:       for k:=1 to totalpuntos do
751:           if Puntos[k].vel<minimo then minimo:=Puntos[k].vel;
752:
753:       for k:=1 to totalpuntos do begin
754:           if abs(maximo-minimo)<>0 then
     aux:=(Puntos[k].vel-minimo)/abs(maximo-minimo)*bmpGraf1.Height*0.95
755:           else aux:=0;
756:           Puntos[k].vel:=bmpGraf1.Height-aux;
757:       end;
758:       for k:=1 to totalpuntos do
759:           Puntos[k].pos:=Puntos[k].pos/totalpuntos*bmpGraf1.Width*0.95;
760:
761:       bmpGraf1.Canvas.Pen.Color:=clwhite;
762:       bmpGraf1.Canvas.Brush.Color:=clwhite;
763:       bmpGraf1.Canvas.Rectangle(0,0,bmpGraf1.Width,bmpGraf1.Height);
764:       bmpGraf1.Canvas.Pen.Color:=clred;
765:       for k:=1 to totalpuntos do begin
766:           bmpGraf1.Canvas.Rectangle(round(Puntos[k].pos)-1,bmpGraf1.Height,
767:                                     round(Puntos[k].pos)+3,round(Puntos[k].vel));
768:
769:       end;
770:       with form1 do
771:            PaintBox7.Canvas.CopyRect(rect(0,0,PaintBox7.Width,PaintBox7.Height),
772:                                      bmpGraf1.Canvas,
773:                                      rect(0,0,bmpGraf1.Width,bmpGraf1.Height));
774: end;
775:
776: procedure GuardaAnalisis2;
777: var  FV:TextFile;
778:      buffer:string;
779:      x,y,k,cont:integer;
780:      d_loc,u_x,u_y,acum,u_squ:real;
781: begin
782:       assignfile(FV,'c:\Analisis2.txt');
783:       rewrite(FV);
784:       buffer:='X              VelProm';
785:       writeln(FV,buffer);
786:       for x:=1 to lx do begin
787:           acum:=0; cont:=0;
788:           for y:=1 to ly do
789:               if (not obst[x,y]) then begin
790:                   d_loc:=0;
791:                   for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
792:                   u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
793:                       -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
794:                   acum:=acum+u_x;
795:                   cont:=cont+1;
796:               end;
797:           acum:=abs(acum/cont);
798:           buffer:=inttostr(x)+'       '+floattostr(acum);
799:           writeln(FV,buffer);
800:       end;
801:       closefile(FV);
```

```
802: end;
803:
804: procedure ResultadoFinal;
805: var x,y,i,n_free:integer;
806:     u_x,d_loc,vel:real;
807: begin
808:      n_free:=0;
809:      u_x:=0;
810:      for x:=1 to lx do
811:          for y:=1 to ly do
812:              if (not obst[x,y]) then begin
813:                  d_loc:=0;
814:                  for i:=0 to 8 do d_loc:=d_loc+node[i,x,y];
815:                  u_x:=u_x+(node[1,x,y]+node[5,x,y]+node[8,x,y]-
816:                           (node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
817:                  n_free:=n_free+1;
818:              end;
819:      vel:=abs(u_x/n_free);
820:      form1.Edit19.Text:=floattostr(vel);
821: end;
822:
823: procedure GuardaPuntos;
824: var  x,y,k,TotalPuntos:integer;
825:      d_loc,u_x,u_y,aux:real;
826:      Datos:array[1..ly]of rec2;
827: begin
828:       for y:=1 to ly do begin
829:           puntos[y].pos:=0;
830:           puntos[y].vel:=0;
831:       end;
832:       x:=posxcorte;//101;
833:       TotalPuntos:=0;
834:       for y:=ly downto 1 do
835:           if (not obst[x,y]) then begin
836:               d_loc:=0;
837:               for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
838:               u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
839:                   -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
840:               TotalPuntos:=TotalPuntos+1;
841:               Datos[TotalPuntos].vel:=u_x;
842:               Datos[TotalPuntos].pos:=TotalPuntos;
843:           end;
844:
845:       if (Datos[TotalPuntos].vel<Datos[1].vel) then begin
846:           Datos[1].vel:=Datos[TotalPuntos-0].vel;
847:           Datos[2].vel:=Datos[TotalPuntos-1].vel;
848:           Datos[3].vel:=Datos[TotalPuntos-2].vel;
849:       end;
850:       if (Datos[1].vel<Datos[TotalPuntos].vel) then begin
851:           Datos[TotalPuntos-0].vel:=Datos[1].vel;
852:           Datos[TotalPuntos-1].vel:=Datos[2].vel;
853:           Datos[TotalPuntos-2].vel:=Datos[3].vel;
854:       end;
855:
856:       k:=0;
857:       for y:=ly downto 1 do
858:           if (not obst[x,y]) then begin
859:               k:=k+1;
860:               Puntos[y].pos:=Datos[k].pos;
861:               Puntos[y].vel:=Datos[k].vel;
862:           end;
863:
864:       pini:=1; y:=1;
865:       while y<=ly do begin
866:             if (puntos[y].vel<>0) then begin pini:=y; y:=ly; end;
867:             y:=y+1;
868:       end;
869:       pfin:=ly; y:=ly;
870:       while y>=1 do begin
871:             if (puntos[y].vel<>0) then begin pfin:=y; y:=1; end;
872:             y:=y-1;
873:       end;
874:
```

```
875:       max:=Puntos[pini].vel;
876:       for y:=pini to pfin do
877:           if Puntos[y].vel>max then max:=Puntos[y].vel;
878:       min:=Puntos[pini].vel;
879:       for y:=pini to pfin do
880:           if Puntos[y].vel<min then min:=Puntos[y].vel;
881:
882:       for y:=pini to pfin do begin
883:           if abs(max-min)<>0 then
884:              aux:=(Puntos[y].vel-min)/abs(max-min)*bmpGraf1.Height*0.95
885:           else aux:=0;
886:           Puntos[y].vel:=bmpGraf1.Height-aux;
887:       end;
888:
889:       for y:=pini to pfin do Puntos[y].pos:=Puntos[y].pos/TotalPuntos*bmpGraf1.Width*0.95;
890: end;
891:
892: procedure GraficaVelProm;
893: var  x,y,k,cont:integer;
894:      d_loc,u_x,u_y,aux,maximo,minimo,acum:real;
895: begin
896:       for x:=1 to lx do begin
897:           acum:=0; cont:=0;
898:           for y:=1 to ly do
899:               if (not obst[x,y]) then begin
900:                   d_loc:=0;
901:                   for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
902:                   u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
903:                       -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
904:                   acum:=acum+u_x;
905:                   cont:=cont+1;
906:               end;
907:           acum:=abs(acum/cont);
908:           Puntos[x].pos:=x;
909:           Puntos[x].vel:=acum;
910:       end;
911:       maximo:=Puntos[1].vel;
912:       for x:=1 to lx do
913:           if Puntos[x].vel>maximo then maximo:=Puntos[x].vel;
914:       for x:=1 to lx do begin
915:           if maximo<>0 then aux:=Puntos[x].vel/maximo*bmpGraf2.Height*0.65
916:           else aux:=0;
917:           Puntos[x].vel:=bmpGraf2.Height-aux;
918:       end;
919:
920:       for x:=1 to lx do Puntos[x].pos:=Puntos[x].pos/lx*bmpGraf2.Width;
921:
922:       bmpGraf2.Canvas.Pen.Color:=clwhite;
923:       bmpGraf2.Canvas.Brush.Color:=clwhite;
924:       bmpGraf2.Canvas.Rectangle(0,0,bmpGraf2.Width,bmpGraf2.Height);
925:       bmpGraf2.Canvas.Brush.Color:=clHighlight;
926:       bmpGraf2.Canvas.Pen.Color:=clHighlight;
927:       bmpGraf2.Canvas.Pen.Width:=1;
928:       for x:=1 to lx-1 do begin
929:           bmpGraf2.Canvas.MoveTo(round(Puntos[x].pos),round(Puntos[x].vel));
930:           bmpGraf2.Canvas.LineTo(round(Puntos[x+1].pos),round(Puntos[x+1].vel));
931:       end;
932:       with form1 do
933:            PaintBox9.Canvas.CopyRect(rect(0,0,PaintBox9.Width,PaintBox9.Height),
934:                                      bmpGraf2.Canvas,
935:                                      rect(0,0,bmpGraf2.Width,bmpGraf2.Height));
936: end;
937:
938: procedure GraficaViscosidad;
939: var  x,y,k,cont,i:integer;
940:      d_loc,u_x,u_y,aux,maximo,minimo,acum,tao:real;
941:      cordenadas:array[1..lx]of rec;
942:      u_xant,u_xdes,u_xant_2,u_xdes_2,aux22:real;
943: begin
944:       for x:=1 to lx do begin
945:           cont:=0; acum:=0;
946:           for y:=1 to ly do
947:               if (not obst[x,y]) then begin
```

```
948:                  d_loc:=0;
949:                  for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
950:                  u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
951:                      -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
952:
953:                  if (y>1) then begin
954:                      if (not obst[x,y-1]) then begin
955:                          d_loc:=0;
956:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-1];
957:                          u_xant:=(n_hlp[1,x,y-1]+n_hlp[5,x,y-1]+n_hlp[8,x,y-1]
958:                                 -(n_hlp[3,x,y-1]+n_hlp[6,x,y-1]+n_hlp[7,x,y-1]))/d_loc;
959:                      end
960:                      else u_xant:=0;
961:                  end
962:                  else u_xant:=0;
963:                  if (y>2) then begin
964:                      if (not obst[x,y-2]) then begin
965:                          d_loc:=0;
966:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-2];
967:                          u_xant_2:=(n_hlp[1,x,y-2]+n_hlp[5,x,y-2]+n_hlp[8,x,y-2]
968:                                   -(n_hlp[3,x,y-2]+n_hlp[6,x,y-2]+n_hlp[7,x,y-2]))/d_loc;
969:                      end
970:                      else u_xant_2:=0;
971:                  end
972:                  else u_xant_2:=0;
973:
974:                  if (y<ly) then begin
975:                      if (not obst[x,y+1]) then begin
976:                          d_loc:=0;
977:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+1];
978:                          u_xdes:=(n_hlp[1,x,y+1]+n_hlp[5,x,y+1]+n_hlp[8,x,y+1]
979:                                 -(n_hlp[3,x,y+1]+n_hlp[6,x,y+1]+n_hlp[7,x,y+1]))/d_loc;
980:                      end
981:                      else u_xdes:=0;
982:                  end
983:                  else u_xdes:=0;
984:                  if (y<(ly-1)) then begin
985:                      if (not obst[x,y+2]) then begin
986:                          d_loc:=0;
987:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+2];
988:                          u_xdes_2:=(n_hlp[1,x,y+2]+n_hlp[5,x,y+2]+n_hlp[8,x,y+2]
989:                                   -(n_hlp[3,x,y+2]+n_hlp[6,x,y+2]+n_hlp[7,x,y+2]))/d_loc;
990:                      end
991:                      else u_xdes_2:=0;
992:                  end
993:                  else u_xdes_2:=0;
994:
995:                  aux22:=abs(-u_xdes_2+8*u_xdes-8*u_xant+u_xant_2)/12;
996:
997:                  if form1.RadioButton1.checked then
998:                     Tao:=omega*aux22;
999:                  if form1.RadioButton2.checked then begin
1000:                    Tao:=ain*power(aux22,nin-1);
1001:                    Tao:=1/(3*Tao+0.5)*aux22;
1002:                 end;
1003:                 cont:=cont+1;
1004:                 acum:=acum+Tao/aux22;
1005:             end;
1006:             cordenadas[x].y:=acum/cont;
1007:             cordenadas[x].x:=x;
1008:       end;
1009:
1010:       maximo:=cordenadas[1].y;
1011:       for x:=1 to lx do
1012:           if cordenadas[x].y>maximo then maximo:=cordenadas[x].y;
1013:       for x:=1 to lx do begin
1014:           if maximo<>0 then aux:=cordenadas[x].y/maximo*bmpGraf4.Height*0.65
1015:           else aux:=0;
1016:           cordenadas[x].y:=bmpGraf4.Height-aux;
1017:       end;
1018:
1019:       for x:=1 to lx do cordenadas[x].x:=cordenadas[x].x/lx*bmpGraf4.Width;
1020:
```

```
1021:       bmpGraf4.Canvas.Pen.Color:=clwhite;
1022:       bmpGraf4.Canvas.Brush.Color:=clwhite;
1023:       bmpGraf4.Canvas.Rectangle(0,0,bmpGraf4.Width,bmpGraf4.Height);
1024:       bmpGraf4.Canvas.Brush.Color:=clHighlight;
1025:       bmpGraf4.Canvas.Pen.Color:=clHighlight;
1026:       bmpGraf4.Canvas.Pen.Width:=1;
1027:       for x:=1 to lx-1 do begin
1028:           bmpGraf4.Canvas.MoveTo(round(cordenadas[x].x),round(cordenadas[x].y));
1029:           bmpGraf4.Canvas.LineTo(round(cordenadas[x+1].x),round(cordenadas[x+1].y));
1030:       end;
1031:       with form1 do
1032:            PaintBox11.Canvas.CopyRect(rect(0,0,PaintBox11.Width,PaintBox11.Height),
1033:                                       bmpGraf4.Canvas,
1034:                                       rect(0,0,bmpGraf4.Width,bmpGraf4.Height));
1035: end;
1036:
1037: procedure GraficaReologica;
1038: var  x,y,k,w,cont,i,Apuntador:integer;
1039:      d_loc,u_x,u_y,aux,maximo,tao,miu:real;
1040:      FV:TextFile;
1041:      buffer:string;
1042:      u_xant,u_xdes,u_xant_2,u_xdes_2,aux22:real;
1043:      recaux:rec2;
1044: begin
1045:       for y:=1 to ly do begin
1046:           Puntos[y].pos:=0;
1047:           Puntos[y].vel:=0;
1048:       end;
1049:
1050:       x:=posxcorte;//101;
1051:       Apuntador:=0;
1052:       for y:=1 to ly do
1053:           if (not obst[x,y]) then begin
1054:                  d_loc:=0;
1055:                  for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
1056:                  u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
1057:                      -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
1058:
1059:                  if (y>1) then begin
1060:                      if (not obst[x,y-1]) then begin
1061:                          d_loc:=0;
1062:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-1];
1063:                          u_xant:=(n_hlp[1,x,y-1]+n_hlp[5,x,y-1]+n_hlp[8,x,y-1]
1064:                                 -(n_hlp[3,x,y-1]+n_hlp[6,x,y-1]+n_hlp[7,x,y-1]))/d_loc;
1065:                      end
1066:                      else u_xant:=0;
1067:                  end
1068:                  else u_xant:=0;
1069:                  if (y>2) then begin
1070:                      if (not obst[x,y-2]) then begin
1071:                          d_loc:=0;
1072:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y-2];
1073:                          u_xant_2:=(n_hlp[1,x,y-2]+n_hlp[5,x,y-2]+n_hlp[8,x,y-2]
1074:                                   -(n_hlp[3,x,y-2]+n_hlp[6,x,y-2]+n_hlp[7,x,y-2]))/d_loc;
1075:                      end
1076:                      else u_xant_2:=0;
1077:                  end
1078:                  else u_xant_2:=0;
1079:
1080:                  if (y<ly) then begin
1081:                      if (not obst[x,y+1]) then begin
1082:                          d_loc:=0;
1083:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+1];
1084:                          u_xdes:=(n_hlp[1,x,y+1]+n_hlp[5,x,y+1]+n_hlp[8,x,y+1]
1085:                                 -(n_hlp[3,x,y+1]+n_hlp[6,x,y+1]+n_hlp[7,x,y+1]))/d_loc;
1086:                      end
1087:                      else u_xdes:=0;
1088:                  end
1089:                  else u_xdes:=0;
1090:                  if (y<(ly-1)) then begin
1091:                      if (not obst[x,y+2]) then begin
1092:                          d_loc:=0;
1093:                          for i:=0 to 8 do d_loc:=d_loc+n_hlp[i,x,y+2];
```

```
1094:                           u_xdes_2:=(n_hlp[1,x,y+2]+n_hlp[5,x,y+2]+n_hlp[8,x,y+2]
1095:                                   -(n_hlp[3,x,y+2]+n_hlp[6,x,y+2]+n_hlp[7,x,y+2]))/d_loc;
1096:                       end
1097:                       else u_xdes_2:=0;
1098:                   end
1099:                   else u_xdes_2:=0;
1100:
1101:                   aux22:=abs(-u_xdes_2+8*u_xdes-8*u_xant+u_xant_2)/12;
1102:
1103:                   if form1.RadioButton1.checked then
1104:                      Tao:=omega*aux22;
1105:                   if form1.RadioButton2.checked then
1106:                      Tao:=ain*power(aux22,nin-1)*aux22;
1107:
1108:                   Apuntador:=Apuntador+1;
1109:                   Puntos[Apuntador].pos:=aux22;
1110:                   Puntos[Apuntador].vel:=Tao;
1111:           end;
1112:
1113:       for k:=1 to Apuntador-1 do
1114:           for w:=k+1 to Apuntador do
1115:               if Puntos[w].vel<Puntos[k].vel then begin
1116:                  recaux:=Puntos[k];
1117:                  Puntos[k]:=Puntos[w];
1118:                  Puntos[w]:=recaux;
1119:               end;
1120:
1121:       maximo:=Puntos[1].vel;
1122:       for k:=1 to Apuntador do
1123:           if Puntos[k].vel>maximo then maximo:=Puntos[k].vel;
1124:
1125:       for k:=1 to Apuntador do begin
1126:           if maximo<>0 then aux:=(Puntos[k].vel/maximo*bmpGraf3.Height)
1127:           else aux:=0;
1128:           Puntos[k].vel:=bmpGraf3.Height-aux;
1129:       end;
1130:
1131:       maximo:=Puntos[1].pos;
1132:       for k:=1 to Apuntador do
1133:           if Puntos[k].pos>maximo then maximo:=Puntos[k].pos;
1134:       for k:=1 to Apuntador do begin
1135:           if maximo<>0 then aux:=(Puntos[k].pos/maximo*bmpGraf3.Width)
1136:           else aux:=0;
1137:           Puntos[k].pos:=aux;
1138:       end;
1139:
1140:       bmpGraf3.Canvas.Pen.Color:=clwhite;
1141:       bmpGraf3.Canvas.Brush.Color:=clwhite;
1142:       bmpGraf3.Canvas.Rectangle(0,0,bmpGraf3.Width,bmpGraf3.Height);
1143:       bmpGraf3.Canvas.Pen.Color:=clblack;
1144:
1145:       x:=posxcorte;//101;
1146:       bmpGraf3.Canvas.Brush.Color:=clHighlight;
1147:       bmpGraf3.Canvas.Pen.Color:=clHighlight;
1148:       bmpGraf3.Canvas.Pen.Width:=1;
1149:
1150:       for k:=1 to Apuntador-1 do begin
1151:           bmpGraf3.Canvas.MoveTo(round(Puntos[k].pos),round(Puntos[k].vel));
1152:           bmpGraf3.Canvas.LineTo(round(Puntos[k+1].pos),round(Puntos[k+1].vel));
1153:       end;
1154:
1155:       with form1 do
1156:            PaintBox10.Canvas.CopyRect(rect(0,0,PaintBox10.Width,PaintBox10.Height),
1157:                                       bmpGraf3.Canvas,
1158:                                       rect(0,0,bmpGraf3.Width,bmpGraf3.Height));
1159: end;
1160:
1161: procedure VelMaxEntySal;
1162: var  x,y,k,pini,pfin:integer;
1163:      d_loc,u_x,maximo,VelEnt,VelSal:real;
1164: begin
1165:       for y:=1 to ly do begin
1166:           puntos[y].pos:=0;
```

```
1167:           puntos[y].vel:=0;
1168:       end;
1169:       x:=1;
1170:       for y:=ly downto 1 do
1171:           if (not obst[x,y]) then begin
1172:               d_loc:=0;
1173:               for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
1174:               u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
1175:                   -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
1176:               Puntos[y].pos:=0;
1177:               Puntos[y].vel:=u_x;
1178:           end;
1179:       pini:=1; y:=1;
1180:       while y<=ly do begin
1181:             if (puntos[y].vel<>0) then begin pini:=y; y:=ly; end;
1182:             y:=y+1;
1183:       end;
1184:       pfin:=ly; y:=ly;
1185:       while y>=1 do begin
1186:             if (puntos[y].vel<>0) then begin pfin:=y; y:=1; end;
1187:             y:=y-1;
1188:       end;
1189:       maximo:=Puntos[pini].vel;
1190:       for y:=pini to pfin do
1191:           if Puntos[y].vel>maximo then maximo:=Puntos[y].vel;
1192:       form1.Edit20.Text:=floattostr(maximo);
1193:       //-----------------------------------
1194:
1195:       for y:=1 to ly do begin
1196:           puntos[y].pos:=0;
1197:           puntos[y].vel:=0;
1198:       end;
1199:       x:=ly;
1200:       for y:=ly downto 1 do
1201:           if (not obst[x,y]) then begin
1202:               d_loc:=0;
1203:               for k:=0 to 8 do d_loc:=d_loc+node[k,x,y];
1204:               u_x:=(node[1,x,y]+node[5,x,y]+node[8,x,y]
1205:                   -(node[3,x,y]+node[6,x,y]+node[7,x,y]))/d_loc;
1206:               Puntos[y].pos:=0;
1207:               Puntos[y].vel:=u_x;
1208:           end;
1209:       pini:=1; y:=1;
1210:       while y<=ly do begin
1211:             if (puntos[y].vel<>0) then begin pini:=y; y:=ly; end;
1212:             y:=y+1;
1213:       end;
1214:       pfin:=ly; y:=ly;
1215:       while y>=1 do begin
1216:             if (puntos[y].vel<>0) then begin pfin:=y; y:=1; end;
1217:             y:=y-1;
1218:       end;
1219:       maximo:=Puntos[pini].vel;
1220:       for y:=pini to pfin do
1221:           if Puntos[y].vel>maximo then maximo:=Puntos[y].vel;
1222:       form1.Edit21.Text:=floattostr(maximo);
1223: end;
1224:
1225: function Calculos(noused:pointer):integer;
1226: var time:integer;
1227: begin
1228:      {assignfile(f1,'c:\result1.txt');
1229:      assignfile(fm1,'c:\resultfm1.dat');
1230:      assignfile(fm2,'c:\resultfm2.dat');
1231:      assignfile(fm3,'c:\resultfm3.dat');
1232:      assignfile(fm4,'c:\resultfm4.dat');}
1233:      assignfile(fm5,'c:\resultfm5.dat');
1234:      {rewrite(f1);
1235:      buffer:='time  velx  vely  press';
1236:      writeln(f1,buffer);
1237:      rewrite(fm1,1); rewrite(fm2,1); rewrite(fm3,1); rewrite(fm4,1);}
1238:      rewrite(fm5,1);
1239:      form1.ProgressBar1.Max:=t_max;
```

```
1240:      form1.ProgressBar1.Min:=1;
1241:      form1.ProgressBar1.Position:=0;
1242:      form1.Timer3.Enabled:=true;
1243:      form1.calculaMatrizDeltas;
1244:
1245:      for time:=1 to t_max do begin
1246:          form1.ProgressBar1.Position:=time;
1247:          //if (time mod (t max div 100))=0 then
1248:          if time>5 then GuardaPuntos;
1249:          if (time >= 10)and((time mod (t_max div 10))=0) then begin
1250:              {GuardaVelVx;
1251:              GuardaPresion(density);
1252:              GuardaVelVxVy;
1253:              GuardaSumaVel;}
1254:              GuardaVectores;
1255:              Check_density(time);
1256:          end;
1257:          Propagate;
1258:          bounceback;
1259:          relaxation(density,omega,time);
1260:          //write velocity1(lx,ly,time,obst,node,density);
1261:          if Abortar then begin
1262:             GuardaAnalisis1;
1263:             GuardaAnalisis2;
1264:             ResultadoFinal;
1265:             GraficaVelProm;
1266:             GraficaViscosidad;
1267:             GraficaReologica;
1268:             VelMaxEntySal;
1269:             exit;
1270:          end;
1271:      end;
1272:      Abortar:=true;
1273:      form1.Timer3.Enabled:=false;
1274:      {closefile(fm1); Apfm1:=0;
1275:      closefile(fm2); Apfm2:=0;
1276:      closefile(fm3); Apfm3:=0;
1277:      closefile(fm4); Apfm4:=0;}
1278:      closefile(fm5); Apfm5:=0;
1279:      {write results(lx,ly,obst,node,density);
1280:      comp rey(lx,ly,obst,node,time,omega,density,r rey);
1281:      closefile(f1);}
1282:
1283:      GuardaAnalisis1;
1284:      //GuardaAnalisis2;
1285:      ResultadoFinal;
1286:      GraficaVelProm;
1287:      GraficaViscosidad;
1288:      GraficaReologica;
1289:      //GraficaAnalisis1;
1290:      VelMaxEntySal;
1291: end;
1292:
1293: procedure InicializaVariables;
1294: var c,x,y:integer;
1295: begin
1296:      for x:=1 to lx do
1297:          for y:=1 to ly do
1298:              for c:=0 to 8 do begin
1299:                  node[c,x,y]:=0;
1300:                  n_hlp[c,x,y]:=0;
1301:              end;
1302:      for x:=1 to lx do
1303:          for y:=1 to ly do obst[x,y]:=false;
1304:
1305:      {for x:=0 to lx-1 do begin
1306:          obst[x+1,1]:=true;
1307:          obst[x+1,ly]:=true;
1308:      end;}
1309: end;
1310:
1311: procedure TForm1.SpeedButton1Click(Sender: TObject);
1312: var  FV:TextFile;
```

```
1313: begin
1314:      abortar:=true;
1315:      if FileExists('c:\Tempo.txt') then begin
1316:         assignfile(FV,'c:\Tempo.txt');
1317:         Erase(FV);
1318:      end;
1319:      sleep(100);
1320:      Application.Terminate;
1321: end;
1322:
1323: procedure tform1.cuadricula;
1324: var x,contx,y,conty:integer;
1325: begin
1326:         bmp.Canvas.Pen.Color:=clred;
1327:         x:=43; contx:=1;
1328:         while contx<=135 do begin
1329:             bmp.Canvas.MoveTo(x,22);
1330:             bmp.Canvas.LineTo(x,429);
1331:             x:=x+3; contx:=contx+1;
1332:         end;
1333:
1334:         y:=24; conty:=1;
1335:         while conty<=135 do begin
1336:             bmp.Canvas.MoveTo(41,y);
1337:             bmp.Canvas.LineTo(448,y);
1338:             y:=y+3; conty:=conty+1;
1339:         end;
1340:         PaintBox1.Canvas.CopyRect(rect(0,0,PaintBox1.Width,PaintBox1.Height),
1341:                                   bmp.Canvas,
1342:                                   rect(0,0,bmp.Width,bmp.Height));
1343: end;
1344:
1345: procedure TForm1.PaintBox1Paint(Sender: TObject);
1346: begin
1347:      bmp.Canvas.CopyRect(rect(0,0,bmp.Width,bmp.Height),
1348:                          image2.Picture.Bitmap.Canvas,
1349:                          rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
1350:      cuadricula;
1351:      Timer6.Enabled:=true;
1352: end;
1353:
1354: procedure TForm1.FormCreate(Sender: TObject);
1355: var st:string;
1356: begin
1357:      bmp:=tbitmap.Create;
1358:      bmp.Width:=PaintBox1.Width;
1359:      bmp.Height:=PaintBox1.Height;
1360:      bmp2:=tbitmap.Create;
1361:      bmp2.Width:=PaintBox1.Width;
1362:      bmp2.Height:=PaintBox1.Height;
1363:      bmpArteriola:=tbitmap.Create;
1364:      bmpArteriola.Width:=PaintBox1.Width;
1365:      bmpArteriola.Height:=PaintBox1.Height;
1366:      PageControl1.ActivePage:=TabSheet1;
1367:      RadioButton2.Checked:=true;
1368:      ScrollBar1.Position:=10;
1369:      Edit8.Text:=inttostr(ScrollBar1.Position);
1370:
1371:      bmp3:=tbitmap.Create;
1372:      bmp3.Width:=PaintBox2.Width;
1373:      bmp3.Height:=PaintBox2.Height;
1374:      ScrollBar2.Max:=130;
1375:      ScrollBar2.Position:=35;
1376:      Edit6.Text:=inttostr(ScrollBar2.Position);
1377:      Timer1.Interval:=ScrollBar2.Max-ScrollBar2.Position+1;
1378:      timer1.Enabled:=true;
1379:      Timer2.Interval:=50;
1380:      timer2.Enabled:=true;
1381:      cont:=0;
1382:      cont2:=0;
1383:
1384:      bmp4:=tbitmap.Create;
1385:      bmp4.Width:=PaintBox3.Width;
```

```
1386:      bmp4.Height:=PaintBox3.Height;
1387:
1388:      bmp5:=tbitmap.Create;
1389:      bmp5.Width:=PaintBox4.Width;
1390:      bmp5.Height:=PaintBox4.Height;
1391:
1392:      ScrollBar3.Position:=160;
1393:      st:=floattostr(ScrollBar3.Position/(ScrollBar3.Max-ScrollBar3.Min)*0.8);
1394:      Edit7.Text:=copy(st,1,3);
1395:      Edit11.Text:=copy(st,1,3);
1396:      Edit22.Text:=copy(st,1,3);
1397:      st:=floattostr((ScrollBar4.Position-ScrollBar4.Min)/(ScrollBar4.Max-ScrollBar4.Min)*4+1);
1398:      Edit24.Text:=copy(st,1,3);
1399:
1400:      st:=floattostr(ScrollBar5.Position/(ScrollBar5.Max-ScrollBar5.Min)*0.8);
1401:      Edit16.Text:=copy(st,1,3);
1402:      st:=floattostr((ScrollBar6.Position-ScrollBar6.Min)/(ScrollBar6.Max-ScrollBar6.Min)*4+1);
1403:      Edit25.Text:=copy(st,1,3);
1404:
1405:      bmpgraf1:=tbitmap.Create;
1406:      bmpgraf1.Width:=PaintBox7.Width;
1407:      bmpgraf1.Height:=PaintBox7.Height;
1408:      Abortar:=false;
1409:      ValorAnterior:=0;
1410:      Diferencia:=0;
1411:      bmpgraf2:=tbitmap.Create;
1412:      bmpgraf2.Width:=PaintBox10.Width;
1413:      bmpgraf2.Height:=PaintBox10.Height;
1414:      bmpgraf3:=tbitmap.Create;
1415:      bmpgraf3.Width:=PaintBox9.Width;
1416:      bmpgraf3.Height:=PaintBox9.Height;
1417:      bmpgraf4:=tbitmap.Create;
1418:      bmpgraf4.Width:=PaintBox11.Width;
1419:      bmpgraf4.Height:=PaintBox11.Height;
1420:
1421:      Timer3.Enabled:=false;
1422:      Timer3.Interval:=100;
1423:      Abortar:=true;
1424:
1425:      Movio1:=true;
1426:      Movio2:=false;
1427:      Movio3:=true;
1428:      Movio4:=false;
1429:
1430:      grafica1;
1431:      velEntrada;
1432:      grafica2;
1433:      velSalida;
1434: end;
1435:
1436: procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
1437: begin
1438:      bmp.Free;
1439:      bmp2.Free;
1440:      bmp3.Free;
1441:      bmp4.Free;
1442:      bmp5.Free;
1443:      bmpgraf1.Free;
1444:      bmpgraf2.Free;
1445:      bmpgraf3.Free;
1446:      bmpgraf4.Free;
1447:      bmpArteriola.Free;
1448: end;
1449:
1450: procedure init_density(density:real);
1451: var x,y:integer;
1452:     t_0,t_1,t_2:real;
1453: begin
1454:     t_0:=density * 4/9;
1455:     t_1:=density / 9;
1456:     t_2:=density /36;
1457:     for x:=1 to lx do
1458:         for y:=1 to ly do begin
```

```
1459:            node[0,x,y]:=t_0;
1460:
1461:            node[1,x,y]:=t_1;
1462:            node[2,x,y]:=t_1;
1463:            node[3,x,y]:=t_1;
1464:            node[4,x,y]:=t_1;
1465:
1466:            node[5,x,y]:=t_2;
1467:            node[6,x,y]:=t_2;
1468:            node[7,x,y]:=t_2;
1469:            node[8,x,y]:=t_2;
1470:        end;
1471: end;
1472:
1473: procedure tform1.iniventanas;
1474: begin
1475:        Memo1.Clear;
1476:        Edit15.Clear;
1477:        Edit17.Clear;
1478:        Edit18.Clear;
1479:        Edit19.Clear;
1480:        Edit20.Clear;
1481:        Edit21.Clear;
1482:        form1.ProgressBar1.Position:=0;
1483:        bmpGraf1.Canvas.Pen.Color:=clwhite;
1484:        bmpGraf1.Canvas.Brush.Color:=clwhite;
1485:        bmpGraf1.Canvas.Rectangle(0,0,bmpGraf1.Width,bmpGraf1.Height);
1486:        PaintBox7.Canvas.CopyRect(rect(0,0,PaintBox7.Width,PaintBox7.Height),
1487:                                  bmpGraf1.Canvas,
1488:                                  rect(0,0,bmpGraf1.Width,bmpGraf1.Height));
1489:
1490:        bmpGraf2.Canvas.Pen.Color:=clwhite;
1491:        bmpGraf2.Canvas.Brush.Color:=clwhite;
1492:        bmpGraf2.Canvas.Rectangle(0,0,bmpGraf2.Width,bmpGraf2.Height);
1493:        PaintBox9.Canvas.CopyRect(rect(0,0,PaintBox9.Width,PaintBox9.Height),
1494:                                  bmpGraf2.Canvas,
1495:                                  rect(0,0,bmpGraf2.Width,bmpGraf2.Height));
1496:
1497:        bmpGraf3.Canvas.Pen.Color:=clwhite;
1498:        bmpGraf3.Canvas.Brush.Color:=clwhite;
1499:        bmpGraf3.Canvas.Rectangle(0,0,bmpGraf3.Width,bmpGraf3.Height);
1500:        PaintBox10.Canvas.CopyRect(rect(0,0,PaintBox10.Width,PaintBox10.Height),
1501:                                   bmpGraf3.Canvas,
1502:                                   rect(0,0,bmpGraf3.Width,bmpGraf3.Height));
1503:
1504:        bmpGraf4.Canvas.Pen.Color:=clwhite;
1505:        bmpGraf4.Canvas.Brush.Color:=clwhite;
1506:        bmpGraf4.Canvas.Rectangle(0,0,bmpGraf4.Width,bmpGraf4.Height);
1507:        PaintBox11.Canvas.CopyRect(rect(0,0,PaintBox11.Width,PaintBox11.Height),
1508:                                   bmpGraf4.Canvas,
1509:                                   rect(0,0,bmpGraf4.Width,bmpGraf4.Height));
1510: end;
1511:
1512: procedure TForm1.SpeedButton4Click(Sender: TObject);
1513: var x,contx,y,conty:integer;
1514:     punto:integer;
1515:     THID:Dword;
1516:     ContObst,TotalPuntosArriba,TotalPuntosAbajo:integer;
1517: begin
1518:      bmpArteriola.Canvas.CopyRect(rect(0,0,bmpArteriola.Width,bmpArteriola.Height),
1519:                                   image2.Picture.Bitmap.Canvas,
1520:
       rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
1521:
1522:      for x:=1 to lx do
1523:          for y:=1 to ly do
1524:              obstBack[x,y]:=false;
1525:
1526:      x:=43; contx:=1;
1527:      while contx<=lx do begin
1528:            y:=24; conty:=1;
1529:            while conty<=ly do begin
1530:                  if bmpArteriola.Canvas.Pixels[x-1,y-1]<RGB(125,125,125) then
```

```
1531:                    obstBack[contx,conty]:=true;
1532:                 conty:=conty+1; y:=y+3;
1533:              end;
1534:              contx:=contx+1; x:=x+3;
1535:        end;
1536:
1537:        InicializaVariables;
1538:
1539:        //if CheckBox1.Checked then begin
1540:           for x:=1 to lx do begin
1541:               ContObst:=0;
1542:               for y:=1 to ly do
1543:                   if obstBack[x,y] then ContObst:=ContObst+1;
1544:               TotalPuntosArriba:=(ContObst div 2);
1545:               TotalPuntosAbajo:=ContObst-TotalPuntosArriba;
1546:               for y:=1 to TotalPuntosArriba do obst[x,y]:=true;
1547:               for y:=1 to TotalPuntosAbajo  do obst[x,ly-y+1]:=true;
1548:           end;
1549:        //end;
1550:
1551:        {if not CheckBox1.Checked then begin
1552:           for x:=1 to lx do
1553:               for y:=1 to ly do obst[x,y]:=obstBack[x,y];
1554:        end;}
1555:
1556:        punto:=0;
1557:        try t_max:=strtoint(Edit2.Text);     except t_max:=0;   punto:=1; end;
1558:        try density:=strtofloat(Edit3.Text); except density:=0; punto:=2; end;
1559:        try omega:=strtofloat(Edit5.Text);   except omega:=0;   punto:=3; end;
1560:        try ain:=strtofloat(Edit10.Text);     except ain:=1;    punto:=4; end;
1561:        try nin:=strtofloat(Edit11.Text);     except nin:=1;    punto:=5; end;
1562:
1563:        try UxE:=strtofloat(Edit9.Text);     except UxE:=0;    punto:=6; end;
1564:        try UyE:=strtofloat(Edit12.Text);     except UyE:=0;    punto:=7; end;
1565:        try UxS:=strtofloat(Edit13.Text);     except UxS:=0;    punto:=8; end;
1566:        try UyS:=strtofloat(Edit14.Text);     except UyS:=0;    punto:=9; end;
1567:
1568:        try FactorAjuste:=strtofloat(Edit1.Text);
1569:        except FactorAjuste:=1;    punto:=10; end;
1570:
1571:        if punto=0 then begin
1572:           iniventanas;
1573:           Diferencia:=0;
1574:           ValorAnterior:=0;
1575:           init_density(density);
1576:           Abortar:=false;
1577:           velEntrada;
1578:           velSalida;
1579:           CreateThread(nil,0,@Calculos,nil,0,THID);
1580:        end
1581:        else showmessage('ERROR EN LA LECTURA DE LOS PARÁMETROS...');
1582: end;
1583:
1584: procedure TForm1.SpeedButton2Click(Sender: TObject);
1585: var x,y,k,w,R,Escala:integer;
1586:     M:tipomat2;
1587: begin
1588:       bmp2.Canvas.pen.Color:=clwhite;
1589:       bmp2.Canvas.Brush.Color:=clwhite;
1590:       bmp2.Canvas.Rectangle(0,0,bmp2.Width,bmp2.Height);
1591:       bmp2.Canvas.CopyRect(rect(0,0,bmp2.Width,bmp2.Height),
1592:                                 image3.Picture.Bitmap.Canvas,
1593:
       rect(0,0,image3.Picture.Bitmap.Width,image3.Picture.Bitmap.Height));
1594:       assignfile(fm5,'c:\resultfm5.dat');
1595:       Reset(fm5,1);
1596:       seek(fm5,Apfm5);
1597:       blockread(fm5,M,SizeOf(M),R);
1598:       Gauge1.MaxValue:=FileSize(fm5);
1599:       Gauge1.Progress:=Apfm5;
1600:       Escala:=ScrollBar1.Position;
1601:       if R>0 then begin
1602:          Apfm5:=Apfm5+SizeOf(M);
```

```
1603:          x:=43;k:=1;
1604:          while k<=lx do begin
1605:                y:=21; w:=1;
1606:                while w<=ly do begin
1607:                      bmp2.Canvas.MoveTo(x-2,y-2);
1608:                      bmp2.Canvas.pen.Color:=clred;
1609:                      bmp2.Canvas.LineTo(x+round(M[k,w].x*Escala),y+round(M[k,w].y*Escala));
1610:                      bmp2.Canvas.pen.Color:=clgreen;
1611:                      bmp2.Canvas.Ellipse(x+round(M[k,w].x*Escala)-2,y+round(M[k,w].y*Escala)-2,
1612:                                          x+round(M[k,w].x*Escala)+2,y+round(M[k,w].y*Escala)+2);
1613:                      w:=w+1; y:=y+3;
1614:                end;
1615:                k:=k+1; x:=x+3;
1616:          end;
1617:          bmp2.Canvas.pen.Width:=1;
1618:          PaintBox1.Canvas.CopyRect(rect(0,0,PaintBox1.Width,PaintBox1.Height),
1619:                                    bmp2.Canvas,
1620:                                    rect(0,0,bmp2.Width,bmp2.Height));
1621:       end
1622:       else Apfm5:=0;
1623:       closefile(fm5);
1624: end;
1625:
1626: procedure TForm1.ScrollBar1Change(Sender: TObject);
1627: begin
1628:      Edit8.Text:=inttostr(ScrollBar1.Position);
1629: end;
1630:
1631: procedure TForm1.SpeedButton3Click(Sender: TObject);
1632: begin
1633:      bmp.Canvas.CopyRect(rect(0,0,bmp.Width,bmp.Height),
1634:                          image2.Picture.Bitmap.Canvas,
1635:                          rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
1636:      cuadricula;
1637: end;
1638:
1639: procedure TForm1.SpeedButton5Click(Sender: TObject);
1640: begin
1641:      PaintBox1.Canvas.CopyRect(rect(0,0,PaintBox1.Width,PaintBox1.Height),
1642:                                bmp2.Canvas,
1643:                                rect(0,0,bmp2.Width,bmp2.Height));
1644: end;
1645:
1646: procedure tform1.grafica1;
1647: var y,limite:integer;
1648:     aux,maximo,tao,miu,K,N:real;
1649: begin
1650:       for y:=1 to ly do begin
1651:           Puntos3[y].x:=0;
1652:           Puntos3[y].y:=0;
1653:       end;
1654:
1655:       if Movio1 then N:=(ScrollBar3.Max+ScrollBar3.Min-ScrollBar3.Position)/10;
1656:       if Movio2 then N:=(ScrollBar4.Max+ScrollBar4.Min-ScrollBar4.Position)/100;
1657:       K:=1;
1658:
1659:       Limite:=ly div 2;
1660:
1661:       for y:=1 to limite do begin
1662:           Tao:=K*power(limite-y+1,N-1);
1663:           Puntos3[y].y:=y;
1664:           Puntos3[y].x:=Tao;
1665:       end;
1666:
1667:       for y:=1 to limite do begin
1668:           Tao:=K*power(y,N-1);
1669:           Puntos3[y+limite].y:=y+limite;
1670:           Puntos3[y+limite].x:=Tao;
1671:       end;
1672:
1673:       limite:=limite*2;
1674:
1675:       maximo:=Puntos3[1].y;
```

```
1676:       for y:=1 to limite do
1677:           if Puntos3[y].y>maximo then maximo:=Puntos3[y].y;
1678:
1679:       for y:=1 to limite do begin
1680:           if maximo<>0 then aux:=(Puntos3[y].y/maximo*(bmp4.Height-1))
1681:           else aux:=0;
1682:           Puntos3[y].y:=aux;
1683:       end;
1684:
1685:       maximo:=Puntos3[1].x;
1686:       for y:=1 to limite do
1687:           if Puntos3[y].x>maximo then maximo:=Puntos3[y].x;
1688:       for y:=1 to limite do begin
1689:           if maximo<>0 then aux:=(Puntos3[y].x/maximo*bmp4.Width)
1690:           else aux:=0;
1691:           Puntos3[y].x:=aux;
1692:       end;
1693:
1694:      bmp4.Canvas.Pen.Color:=clwhite;
1695:      bmp4.Canvas.Brush.Color:=clwhite;
1696:      bmp4.Canvas.Rectangle(0,0,bmp4.Width,bmp4.Height);
1697:      bmp4.Canvas.Pen.Color:=clred;
1698:      bmp4.Canvas.Pen.Width:=1;
1699:
1700:      for y:=1 to limite-1 do begin
1701:           bmp4.Canvas.MoveTo(round(Puntos3[y].x),round(Puntos3[y].y));
1702:           bmp4.Canvas.LineTo(round(Puntos3[y+1].x),round(Puntos3[y+1].y));
1703:      end;
1704:
1705:      bmp4.Canvas.CopyRect(rect(0,0,bmp4.Width,bmp4.Height),
1706:                           bmp4.Canvas,
1707:                           rect(bmp4.Width-1,0,-1,bmp4.Height));
1708:
1709:      PaintBox3.Canvas.CopyRect(rect(0,0,PaintBox3.Width,PaintBox3.Height),
1710:                                bmp4.Canvas,
1711:                                rect(0,0,bmp4.Width,bmp4.Height));
1712: end;
1713:
1714: procedure tform1.grafica2;
1715: var y,limite:integer;
1716:     aux,maximo,tao,miu,K,N:real;
1717: begin
1718:      for y:=1 to ly do begin
1719:           Puntos4[y].x:=0;
1720:           Puntos4[y].y:=0;
1721:       end;
1722:
1723:       {if Movio3 then nin:=(ScrollBar5.Max+ScrollBar5.Min-ScrollBar5.Position)/10;
1724:       if Movio4 then nin:=(ScrollBar6.Max+ScrollBar6.Min-ScrollBar6.Position)/100;}
1725:       if Movio1 then N:=(ScrollBar3.Max+ScrollBar3.Min-ScrollBar3.Position)/10;
1726:       if Movio2 then N:=(ScrollBar4.Max+ScrollBar4.Min-ScrollBar4.Position)/100;
1727:       K:=1;
1728:
1729:       Limite:=ly div 2;
1730:
1731:       for y:=1 to limite do begin
1732:           Tao:=K*power(limite-y+1,N-1);
1733:           Puntos4[y].y:=y;
1734:           Puntos4[y].x:=Tao;
1735:       end;
1736:
1737:       for y:=1 to limite do begin
1738:           Tao:=K*power(y,N-1);
1739:           Puntos4[y+limite].y:=y+limite;
1740:           Puntos4[y+limite].x:=Tao;
1741:       end;
1742:
1743:       limite:=limite*2;
1744:
1745:       maximo:=Puntos4[1].y;
1746:       for y:=1 to limite do
1747:           if Puntos4[y].y>maximo then maximo:=Puntos4[y].y;
1748:
```

```
1749:       for y:=1 to limite do begin
1750:           if maximo<>0 then aux:=(Puntos4[y].y/maximo*(bmp5.Height-1))
1751:           else aux:=0;
1752:           Puntos4[y].y:=aux;
1753:       end;
1754:
1755:       maximo:=Puntos4[1].x;
1756:       for y:=1 to limite do
1757:           if Puntos4[y].x>maximo then maximo:=Puntos4[y].x;
1758:       for y:=1 to limite do begin
1759:           if maximo<>0 then aux:=(Puntos4[y].x/maximo*bmp5.Width)
1760:           else aux:=0;
1761:           Puntos4[y].x:=aux;
1762:       end;
1763:
1764:      bmp5.Canvas.Pen.Color:=clwhite;
1765:      bmp5.Canvas.Brush.Color:=clwhite;
1766:      bmp5.Canvas.Rectangle(0,0,bmp5.Width,bmp5.Height);
1767:      bmp5.Canvas.Pen.Color:=clred;
1768:      bmp5.Canvas.Pen.Width:=1;
1769:
1770:      for y:=1 to limite-1 do begin
1771:           bmp5.Canvas.MoveTo(round(Puntos4[y].x),round(Puntos4[y].y));
1772:           bmp5.Canvas.LineTo(round(Puntos4[y+1].x),round(Puntos4[y+1].y));
1773:      end;
1774:
1775:      bmp5.Canvas.CopyRect(rect(0,0,bmp5.Width,bmp5.Height),
1776:                           bmp5.Canvas,
1777:                           rect(bmp5.Width-1,0,-1,bmp5.Height));
1778:
1779:      PaintBox4.Canvas.CopyRect(rect(0,0,PaintBox4.Width,PaintBox4.Height),
1780:                                bmp5.Canvas,
1781:                                rect(0,0,bmp5.Width,bmp5.Height));
1782: end;
1783:
1784: procedure TForm1.ScrollBar2Change(Sender: TObject);
1785: begin
1786:    if Abortar then begin
1787:      Timer1.Interval:=ScrollBar2.Max-ScrollBar2.Position+1;
1788:      Edit6.Text:=inttostr(ScrollBar2.Position);
1789:      Timer4.Enabled:=true;
1790:      Timer5.Enabled:=true;
1791:    end;
1792: end;
1793:
1794: procedure TForm1.PaintBox3Paint(Sender: TObject);
1795: begin
1796:      PaintBox3.Canvas.CopyRect(rect(0,0,PaintBox3.Width,PaintBox3.Height),
1797:                                bmp4.Canvas,
1798:                                rect(0,0,bmp4.Width,bmp4.Height));
1799: end;
1800:
1801: procedure CalculaNin;
1802: var st:string;
1803: begin
1804:    with form1 do begin
1805:      if Movio1 then st:=floattostr(ScrollBar3.Position/(ScrollBar3.Max-ScrollBar3.Min)*0.8);
1806:      if Movio2 then
       st:=floattostr((ScrollBar4.Position-ScrollBar4.Min)/(ScrollBar4.Max-ScrollBar4.Min)*4+1);
1807:      Edit22.Text:=copy(st,1,3);
1808:      Edit11.Text:=copy(st,1,3);
1809:    end;
1810: end;
1811:
1812: procedure TForm1.ScrollBar3Change(Sender: TObject);
1813: var st:string;
1814: begin
1815:    if Abortar then begin
1816:       InactivaVentanas;
1817:       Movio2:=false;
1818:       Movio1:=true;
1819:       ScrollBar4.Position:=ScrollBar4.Min;
1820:       st:=floattostr(ScrollBar3.Position/(ScrollBar3.Max-ScrollBar3.Min)*0.8);
```

```
1821:       Edit7.Text:=copy(st,1,3);
1822:       CalculaNin;
1823:       grafica1;
1824:       Timer4.Enabled:=true;
1825:       //-------------------
1826:       Movio4:=false;
1827:       Movio3:=true;
1828:       Edit16.Text:=copy(st,1,3);
1829:       grafica2;
1830:       Timer5.Enabled:=true;
1831:    end;
1832: end;
1833:
1834: procedure TForm1.SpeedButton6Click(Sender: TObject);
1835: begin
1836:      if timer1.Enabled=true then timer1.Enabled:=false
1837:      else timer1.Enabled:=true;
1838: end;
1839:
1840: procedure TForm1.ScrollBar5Change(Sender: TObject);
1841: var st:string;
1842: begin
1843:    {if Abortar then begin
1844:      Movio4:=false;
1845:      Movio3:=true;
1846:      ScrollBar6.Position:=ScrollBar6.Min;
1847:      st:=floattostr(ScrollBar5.Position/(ScrollBar5.Max-ScrollBar5.Min)*0.8);
1848:      Edit16.Text:=copy(st,1,3);
1849:      grafica2;
1850:      Timer5.Enabled:=true;
1851:    end;}
1852: end;
1853:
1854: procedure TForm1.PaintBox4Paint(Sender: TObject);
1855: begin
1856:      PaintBox4.Canvas.CopyRect(rect(0,0,PaintBox4.Width,PaintBox4.Height),
1857:                                bmp5.Canvas,
1858:                                rect(0,0,bmp5.Width,bmp5.Height));
1859: end;
1860:
1861: procedure TForm1.PaintBox2Paint(Sender: TObject);
1862: begin
1863:      PaintBox2.Canvas.CopyRect(rect(0,0,PaintBox2.Width,PaintBox2.Height),
1864:                                bmp3.Canvas,
1865:                                rect(0,0,bmp3.Width,bmp3.Height));
1866: end;
1867:
1868: procedure TForm1.PaintBox5Paint(Sender: TObject);
1869: begin
1870:      PaintBox5.Canvas.CopyRect(rect(0,0,PaintBox5.Width,PaintBox5.Height),
1871:                                bmp3.Canvas,
1872:                                rect(0,0,bmp3.Width,bmp3.Height));
1873: end;
1874:
1875: procedure TForm1.Timer1Timer(Sender: TObject);
1876: begin
1877:       cont:=cont+1;
1878:       if cont>234 then cont:=8;
1879:       if fileExists('c:\BMPs1\Imagen'+inttostr(cont)+'.bmp') then begin
1880:          bmp3.LoadFromFile('c:\BMPs1\Imagen'+inttostr(cont)+'.bmp');
1881:          PaintBox2.Canvas.CopyRect(rect(0,0,PaintBox2.Width,PaintBox2.Height),
1882:                                    bmp3.Canvas,
1883:                                    rect(0,0,bmp3.Width,bmp3.Height));
1884:       end;
1885: end;
1886:
1887: procedure tform1.velEntrada;
1888: var posx,y,x,k,cont1,cont2,cont3,contx,conty:integer;
1889:     delta,posy,mayor,mayor2:real;
1890:     VelNodo:array[1..150]of real;
1891:     st:string;
1892:     VelEntDefinida:real;
1893: begin
```

```
1894:      cont1:=0;
1895:      for y:=1 to bmp4.Height-1 do begin
1896:          posx:=0;
1897:          for x:=0 to bmp4.Width do begin
1898:              if bmp4.Canvas.Pixels[x,y]=clred then posx:=x;
1899:          end;
1900:          cont1:=cont1+1;
1901:          VelNodo[cont1]:=posx;
1902:      end;
1903:
1904:      for k:=1 to (cont1 div 2) do
1905:          VelNodo[cont1-k+1]:=VelNodo[k];
1906:
1907:      Memo2.Clear;
1908:      for k:=1 to cont1 do
1909:          Memo2.Lines.Add(inttostr(round(VelNodo[k])));
1910:
1911:      {bmpArteriola.Canvas.CopyRect(rect(0,0,bmpArteriola.Width,bmpArteriola.Height),
1912:                                    image2.Picture.Bitmap.Canvas,
1913:
     rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
1914:      InicializaVariables;
1915:      x:=43; contx:=1;
1916:      while contx<=lx do begin
1917:            y:=24; conty:=1;
1918:            while conty<=ly do begin
1919:                  if bmpArteriola.Canvas.Pixels[x-1,y-1]<RGB(125,125,125) then
1920:                     obst[contx,conty]:=true;
1921:                  conty:=conty+1; y:=y+3;
1922:            end;
1923:            contx:=contx+1; x:=x+3;
1924:      end;
1925:      cont2:=0;
1926:      for y:=1 to ly do
1927:          if not obst[1,y] then cont2:=cont2+1;}
1928:
1929:      // Normalizar el vector
1930:      mayor:=VelNodo[1];
1931:      for k:=1 to cont1 do
1932:          if VelNodo[k]>mayor then mayor:=VelNodo[k];
1933:
1934:      VelEntDefinida:=strtofloat(Edit26.Text)/10;
1935:      for k:=1 to cont1 do begin
1936:          VelNodo[k]:=VelNodo[k]/mayor*VelEntDefinida;
1937:      end;
1938:      Edit9.Text:=Edit26.Text;
1939:      Memo4.Clear;
1940:      {delta:=cont1/cont2;
1941:      PosY:=delta/2; cont3:=0;
1942:      while PosY<=cont1 do begin
1943:            cont3:=cont3+1;
1944:            Memo4.Lines.Add(inttostr(cont3)+'.- '+floattostr(VelNodo[round(PosY)]));
1945:            PosY:=PosY+delta;
1946:      end;}
1947:      for k:=1 to cont1 do begin
1948:          st:=floattostr(VelNodo[k]);
1949:          st:=copy(st,1,7);
1950:          PerfilVelEnt[k]:=strtofloat(st);
1951:          //PerfilVelEnt[k]:=UxE;
1952:          Memo4.Lines.Add(inttostr(k)+'.- '+st);
1953:      end;
1954: end;
1955:
1956: procedure tform1.VelSalida;
1957: var posx,y,x,k,cont1,cont2,cont3,contx,conty:integer;
1958:     delta,posy,mayor,mayor2:real;
1959:     VelNodo:array[1..150]of real;
1960:     st:string;
1961:     VelSalDefinida:real;
1962: begin
1963:      cont1:=0;
1964:      for y:=1 to bmp5.Height-1 do begin
1965:          posx:=0;
```

```
1966:          for x:=0 to bmp5.Width do begin
1967:              if bmp5.Canvas.Pixels[x,y]=clred then posx:=x;
1968:          end;
1969:          cont1:=cont1+1;
1970:          VelNodo[cont1]:=posx;
1971:      end;
1972:
1973:      for k:=1 to (cont1 div 2) do
1974:          VelNodo[cont1-k+1]:=VelNodo[k];
1975:
1976:      Memo3.Clear;
1977:      for k:=1 to cont1 do
1978:          Memo3.Lines.Add(inttostr(round(VelNodo[k])));
1979:
1980:      {bmpArteriola.Canvas.CopyRect(rect(0,0,bmpArteriola.Width,bmpArteriola.Height),
1981:                                    image2.Picture.Bitmap.Canvas,
1982:
       rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
1983:      InicializaVariables;
1984:      x:=43; contx:=1;
1985:      while contx<=lx do begin
1986:            y:=24; conty:=1;
1987:            while conty<=ly do begin
1988:                  if bmpArteriola.Canvas.Pixels[x-1,y-1]<RGB(125,125,125) then
1989:                     obst[contx,conty]:=true;
1990:                  conty:=conty+1; y:=y+3;
1991:            end;
1992:            contx:=contx+1; x:=x+3;
1993:      end;
1994:
1995:      cont2:=0;
1996:      for y:=1 to ly do
1997:          if not obst[ly,y] then cont2:=cont2+1;}
1998:
1999:      // Normalizar el vector
2000:      mayor:=VelNodo[1];
2001:      for k:=1 to cont1 do
2002:          if VelNodo[k]>mayor then mayor:=VelNodo[k];
2003:
2004:      VelSalDefinida:=strtofloat(Edit28.Text)/10;
2005:      for k:=1 to cont1 do begin
2006:          VelNodo[k]:=VelNodo[k]/mayor*VelSalDefinida;
2007:      end;
2008:      Edit13.Text:=Edit28.Text;
2009:      Memo5.Clear;
2010:      {delta:=cont1/cont2;
2011:      PosY:=delta/2; cont3:=0;
2012:      while PosY<=cont1 do begin
2013:            cont3:=cont3+1;
2014:            Memo5.Lines.Add(inttostr(cont3)+'.- '+floattostr(VelNodo[round(PosY)]));
2015:            PosY:=PosY+delta;
2016:      end;}
2017:      for k:=1 to cont1 do begin
2018:          st:=floattostr(VelNodo[k]);
2019:          st:=copy(st,1,7);
2020:          PerfilVelSal[k]:=strtofloat(st);
2021:          //PerfilVelSal[k]:=UxS;
2022:          Memo5.Lines.Add(inttostr(k)+'.- '+st);
2023:      end;
2024: end;
2025:
2026: procedure TForm1.Timer2Timer(Sender: TObject);
2027: begin
2028:       cont2:=cont2+1;
2029:       if cont2>300 then cont2:=8;
2030:       if fileExists('c:\BMPs2\Imagen'+inttostr(cont2)+'.bmp') then begin
2031:          bmp3.LoadFromFile('c:\BMPs2\Imagen'+inttostr(cont2)+'.bmp');
2032:          PaintBox5.Canvas.CopyRect(rect(0,0,PaintBox5.Width,PaintBox5.Height),
2033:                                    bmp3.Canvas,
2034:                                    rect(0,0,bmp3.Width,bmp3.Height));
2035:          PaintBox6.Canvas.CopyRect(rect(0,0,PaintBox6.Width,PaintBox6.Height),
2036:                                    bmp3.Canvas,
2037:                                    rect(0,0,bmp3.Width,bmp3.Height));
```

```
2038:          PaintBox8.Canvas.CopyRect(rect(0,0,PaintBox8.Width,PaintBox8.Height),
2039:                                    bmp3.Canvas,
2040:                                    rect(0,0,bmp3.Width,bmp3.Height));
2041:       end;
2042: end;
2043:
2044: procedure TForm1.PaintBox7Paint(Sender: TObject);
2045: begin
2046:      PaintBox7.Canvas.CopyRect(rect(0,0,PaintBox7.Width,PaintBox7.Height),
2047:                                bmpGraf1.Canvas,
2048:                                rect(0,0,bmpGraf1.Width,bmpGraf1.Height));
2049: end;
2050:
2051: procedure TForm1.SpeedButton9Click(Sender: TObject);
2052: begin
2053:      Abortar:=true;
2054:      Timer3.Enabled:=false;
2055: end;
2056:
2057: procedure TForm1.PaintBox10Paint(Sender: TObject);
2058: begin
2059:      PaintBox10.Canvas.CopyRect(rect(0,0,PaintBox10.Width,PaintBox10.Height),
2060:                                 bmpGraf3.Canvas,
2061:                                 rect(0,0,bmpGraf3.Width,bmpGraf3.Height));
2062: end;
2063:
2064: procedure TForm1.PaintBox9Paint(Sender: TObject);
2065: begin
2066:     PaintBox9.Canvas.CopyRect(rect(0,0,PaintBox9.Width,PaintBox9.Height),
2067:                               bmpGraf2.Canvas,
2068:                               rect(0,0,bmpGraf2.Width,bmpGraf2.Height));
2069: end;
2070:
2071: procedure calculadiametro(x:integer;var cont:integer);
2072: var y,conty:integer;
2073: begin
2074:      with form1 do
2075:      bmpArteriola.Canvas.CopyRect(rect(0,0,bmpArteriola.Width,bmpArteriola.Height),
2076:                                   image2.Picture.Bitmap.Canvas,
2077:
       rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
2078:      {x:=x*3;} cont:=0;
2079:      y:=24;   conty:=1;
2080:      while conty<=ly do begin
2081:            if bmpArteriola.Canvas.Pixels[x-1,y-1]>=RGB(125,125,125) then
2082:               cont:=cont+1;
2083:            y:=y+3; conty:=conty+1;
2084:      end;
2085: end;
2086:
2087: procedure TForm1.SpeedButton10Click(Sender: TObject);
2088: Var diametro:integer;
2089: begin
2090:    if Abortar then begin
2091:      if OpenPictureDialog1.Execute then begin
2092:         Image2.Picture.LoadFromFile(OpenPictureDialog1.FileName);
2093:         bmp.Canvas.CopyRect(rect(0,0,bmp.Width,bmp.Height),
2094:                             image2.Picture.Bitmap.Canvas,
2095:                             rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
2096:         cuadricula;
2097:         Timer4.Enabled:=true;
2098:         Timer5.Enabled:=true;
2099:         Timer6.Enabled:=true;
2100:         calculadiametro(ScrollBar7.Position,diametro);
2101:         Edit23.Text:=inttostr(diametro);
2102:      end;
2103:    end;
2104: end;
2105:
2106: procedure TForm1.SpeedButton11Click(Sender: TObject);
2107: begin
2108:      iniventanas;
2109: end;
```

```
2110:
2111: procedure TForm1.Timer3Timer(Sender: TObject);
2112: var y:integer;
2113: begin
2114:    Try
2115:       bmpGraf1.Canvas.Pen.Color:=clwhite;
2116:       bmpGraf1.Canvas.Brush.Color:=clwhite;
2117:       bmpGraf1.Canvas.Rectangle(0,0,bmpGraf1.Width,bmpGraf1.Height);
2118:       bmpGraf1.Canvas.Pen.Color:=clred;
2119:       for y:=pini to pfin do begin
2120:           bmpGraf1.Canvas.Rectangle(round(Puntos[y].pos)-1,bmpGraf1.Height,
2121:                                     round(Puntos[y].pos)+3,round(Puntos[y].vel));
2122:
2123:       end;
2124:       with form1 do
2125:            PaintBox7.Canvas.CopyRect(rect(0,0,PaintBox7.Width,PaintBox7.Height),
2126:                                      bmpGraf1.Canvas,
2127:                                      rect(0,0,bmpGraf1.Width,bmpGraf1.Height));
2128:
2129:       form1.Edit15.Text:=floattostr(max*10);
2130:       form1.Edit17.Text:=floattostr(ValorAnterior*10);
2131:       Diferencia:=max-ValorAnterior;
2132:       form1.Edit18.Text:=floattostr(Diferencia*10);
2133:       ValorAnterior:=max;
2134:    Except sleep(10); end;
2135: end;
2136:
2137: procedure TForm1.SpeedButton12Click(Sender: TObject);
2138: begin
2139:      GraficaAnalisis1;
2140: end;
2141:
2142: procedure TForm1.Timer4Timer(Sender: TObject);
2143: begin
2144:      velEntrada;
2145:      Timer4.Enabled:=false;
2146: end;
2147:
2148: procedure TForm1.Timer5Timer(Sender: TObject);
2149: begin
2150:      velSalida;
2151:      Timer5.Enabled:=false;
2152: end;
2153:
2154: procedure TForm1.SpeedButton14Click(Sender: TObject);
2155: begin
2156:      Try GraficaVelProm;    Except end;
2157:      Try GraficaViscosidad; Except end;
2158: end;
2159:
2160: procedure TForm1.SpeedButton13Click(Sender: TObject);
2161: begin
2162:      Try GraficaReologica; Except end;
2163: end;
2164:
2165: procedure TForm1.ScrollBar4Change(Sender: TObject);
2166: var st:string;
2167: begin
2168:    if Abortar then begin
2169:       InactivaVentanas;
2170:       Movio1:=false;
2171:       Movio2:=true;
2172:       ScrollBar3.Position:=ScrollBar3.Max;
2173:       st:=floattostr((ScrollBar4.Position-ScrollBar4.Min)/(ScrollBar4.Max-ScrollBar4.Min)*4+1);
2174:       Edit24.Text:=copy(st,1,3);
2175:       CalculaNin;
2176:       grafica1;
2177:       Timer4.Enabled:=true;
2178:       //-------------------
2179:       Movio3:=false;
2180:       Movio4:=true;
2181:       Edit25.Text:=copy(st,1,3);
2182:       grafica2;
```

```
2183:       Timer5.Enabled:=true;
2184:    end;
2185: end;
2186:
2187: procedure TForm1.ScrollBar6Change(Sender: TObject);
2188: var st:string;
2189: begin
2190:    {if Abortar then begin
2191:      Movio3:=false;
2192:      Movio4:=true;
2193:      ScrollBar5.Position:=ScrollBar5.Max;
2194:      st:=floattostr((ScrollBar6.Position-ScrollBar6.Min)/(ScrollBar6.Max-ScrollBar6.Min)*4+1);
2195:      Edit25.Text:=copy(st,1,3);
2196:      grafica2;
2197:      Timer5.Enabled:=true;
2198:    end;}
2199: end;
2200:
2201: procedure tform1.GraficaPlanaEntrada;
2202: var y,limite:integer;
2203:     aux,maximo:real;
2204: begin
2205:       for y:=1 to ly do begin
2206:           Puntos3[y].x:=0;
2207:           Puntos3[y].y:=0;
2208:       end;
2209:       limite:=ly;
2210:       for y:=1 to limite do begin
2211:           Puntos3[y].y:=y-1;
2212:           Puntos3[y].x:=lx;
2213:       end;
2214:
2215:       maximo:=Puntos3[1].y;
2216:       for y:=1 to limite do
2217:           if Puntos3[y].y>maximo then maximo:=Puntos3[y].y;
2218:
2219:       for y:=1 to limite do begin
2220:           if maximo<>0 then aux:=(Puntos3[y].y/maximo*bmp4.Height)
2221:           else aux:=0;
2222:           Puntos3[y].y:=aux;
2223:       end;
2224:
2225:       maximo:=Puntos3[1].x;
2226:       for y:=1 to limite do
2227:           if Puntos3[y].x>maximo then maximo:=Puntos3[y].x;
2228:       for y:=1 to limite do begin
2229:           if maximo<>0 then aux:=(Puntos3[y].x/maximo)
2230:           else aux:=0;
2231:           Puntos3[y].x:=aux;
2232:       end;
2233:
2234:      bmp4.Canvas.Pen.Color:=clwhite;
2235:      bmp4.Canvas.Brush.Color:=clwhite;
2236:      bmp4.Canvas.Rectangle(0,0,bmp4.Width,bmp4.Height);
2237:      bmp4.Canvas.Pen.Color:=clred;
2238:      bmp4.Canvas.Pen.Width:=1;
2239:
2240:      for y:=1 to limite-1 do begin
2241:           bmp4.Canvas.MoveTo(round(Puntos3[y].x),round(Puntos3[y].y));
2242:           bmp4.Canvas.LineTo(round(Puntos3[y+1].x),round(Puntos3[y+1].y));
2243:      end;
2244:
2245:      bmp4.Canvas.CopyRect(rect(0,0,bmp4.Width,bmp4.Height),
2246:                           bmp4.Canvas,
2247:                           rect(bmp4.Width-1,0,-1,bmp4.Height));
2248:
2249:      PaintBox3.Canvas.CopyRect(rect(0,0,PaintBox3.Width,PaintBox3.Height),
2250:                                bmp4.Canvas,
2251:                                rect(0,0,bmp4.Width,bmp4.Height));
2252: end;
2253:
2254: procedure tform1.GraficaPlanaSalida;
2255: var y,limite:integer;
```

```
2256:     aux,maximo:real;
2257: begin
2258:       for y:=1 to ly do begin
2259:           Puntos4[y].x:=0;
2260:           Puntos4[y].y:=0;
2261:       end;
2262:       limite:=ly;
2263:       for y:=1 to limite do begin
2264:           Puntos4[y].y:=y-1;
2265:           Puntos4[y].x:=lx;
2266:       end;
2267:
2268:       maximo:=Puntos4[1].y;
2269:       for y:=1 to limite do
2270:           if Puntos4[y].y>maximo then maximo:=Puntos4[y].y;
2271:
2272:       for y:=1 to limite do begin
2273:           if maximo<>0 then aux:=(Puntos4[y].y/maximo*bmp5.Height)
2274:           else aux:=0;
2275:           Puntos4[y].y:=aux;
2276:       end;
2277:
2278:       maximo:=Puntos4[1].x;
2279:       for y:=1 to limite do
2280:           if Puntos4[y].x>maximo then maximo:=Puntos4[y].x;
2281:       for y:=1 to limite do begin
2282:           if maximo<>0 then aux:=(Puntos4[y].x/maximo)
2283:           else aux:=0;
2284:           Puntos4[y].x:=aux;
2285:       end;
2286:
2287:       bmp5.Canvas.Pen.Color:=clwhite;
2288:       bmp5.Canvas.Brush.Color:=clwhite;
2289:       bmp5.Canvas.Rectangle(0,0,bmp5.Width,bmp5.Height);
2290:       bmp5.Canvas.Pen.Color:=clred;
2291:       bmp5.Canvas.Pen.Width:=1;
2292:
2293:       for y:=1 to limite-1 do begin
2294:           bmp5.Canvas.MoveTo(round(Puntos4[y].x),round(Puntos4[y].y));
2295:           bmp5.Canvas.LineTo(round(Puntos4[y+1].x),round(Puntos4[y+1].y));
2296:       end;
2297:
2298:       bmp5.Canvas.CopyRect(rect(0,0,bmp5.Width,bmp5.Height),
2299:                            bmp5.Canvas,
2300:                            rect(bmp5.Width-1,0,-1,bmp5.Height));
2301:
2302:       PaintBox4.Canvas.CopyRect(rect(0,0,PaintBox4.Width,PaintBox4.Height),
2303:                                 bmp5.Canvas,
2304:                                 rect(0,0,bmp5.Width,bmp5.Height));
2305: end;
2306:
2307: procedure tform1.InactivaVentanas;
2308: begin
2309:      Edit22.Color:=clYellow;
2310:      Edit22.ReadOnly:=true;
2311:      Edit11.Color:=clYellow;
2312:      Edit11.ReadOnly:=true;
2313: end;
2314:
2315: procedure tform1.ActivaVentanas;
2316: begin
2317:      Edit22.Color:=clInfoBk;
2318:      Edit22.ReadOnly:=false;
2319:      Edit11.Color:=clInfoBk;
2320:      Edit11.ReadOnly:=false;
2321: end;
2322:
2323: procedure TForm1.SpeedButton15Click(Sender: TObject);
2324: var st:string;
2325: begin
2326:    if Abortar then begin
2327:       GraficaPlanaEntrada;
2328:       Timer4.Enabled:=true;
```

```
2329:       GraficaPlanaSalida;
2330:       Timer5.Enabled:=true;
2331:       ActivaVentanas;
2332:    end;
2333: end;
2334:
2335: procedure TForm1.Edit22Change(Sender: TObject);
2336: begin
2337:      Edit11.Text:=Edit22.Text;
2338: end;
2339:
2340: procedure TForm1.Edit11Change(Sender: TObject);
2341: begin
2342:      Edit22.Text:=Edit11.Text;
2343: end;
2344:
2345: procedure TForm1.PaintBox11Paint(Sender: TObject);
2346: begin
2347:     PaintBox11.Canvas.CopyRect(rect(0,0,PaintBox11.Width,PaintBox11.Height),
2348:                                bmpGraf4.Canvas,
2349:                                rect(0,0,bmpGraf4.Width,bmpGraf4.Height));
2350: end;
2351:
2352: procedure tform1.PuntodeCorte(posicion:integer);
2353: var y,posyarr,posyaba:integer;
2354: begin
2355:      bmp.Canvas.pen.Width:=1;
2356:      bmp.Canvas.CopyRect(rect(0,0,bmp.Width,bmp.Height),
2357:                          image2.Picture.Bitmap.Canvas,
2358:                          rect(0,0,image2.Picture.Bitmap.Width,image2.Picture.Bitmap.Height));
2359:      posxcorte:=round((ScrollBar7.Position)/(446-42)*135-13);
2360:      y:=23;
2361:      while bmp.Canvas.Pixels[posicion,y]<RGB(125,125,125) do y:=y+1;
2362:      posyarr:=y;
2363:      y:=425;
2364:      while bmp.Canvas.Pixels[posicion,y]<RGB(125,125,125) do y:=y-1;
2365:      posyaba:=y+1;
2366:
2367:      cuadricula;
2368:      bmp.Canvas.pen.Width:=2;
2369:      bmp.Canvas.pen.Color:=clNavy;
2370:      bmp.Canvas.Brush.Color:=clNavy;
2371:      bmp.Canvas.MoveTo(posicion,15);
2372:      bmp.Canvas.LineTo(posicion,437);
2373:
2374:      bmp.Canvas.MoveTo(posicion-15,posyarr);
2375:      bmp.Canvas.LineTo(posicion+15,posyarr);
2376:      bmp.Canvas.MoveTo(posicion-15,posyaba);
2377:      bmp.Canvas.LineTo(posicion+15,posyaba);
2378:
2379:      PaintBox1.Canvas.CopyRect(rect(0,0,PaintBox1.Width,PaintBox1.Height),
2380:                                bmp.Canvas,
2381:                                rect(0,0,bmp.Width,bmp.Height));
2382:      bmp.Canvas.pen.Width:=1;
2383: end;
2384:
2385: procedure TForm1.ScrollBar7Change(Sender: TObject);
2386: var diametro:integer;
2387: begin
2388:    if Abortar then begin
2389:      PuntodeCorte(ScrollBar7.Position);
2390:      Edit4.Text:=inttostr(round((ScrollBar7.Position)/(446-42)*135-13));
2391:      calculadiametro(ScrollBar7.Position,diametro);
2392:      Edit23.Text:=inttostr(diametro);
2393:    end;
2394: end;
2395:
2396: procedure TForm1.FormShow(Sender: TObject);
2397: begin
2398:      ScrollBar7.Position:=293;//342;
2399:      Timer6.Enabled:=true;
2400: end;
2401:
```

```
2402: procedure TForm1.Timer6Timer(Sender: TObject);
2403: begin
2404:      PuntodeCorte(ScrollBar7.Position);
2405:      Edit4.Text:=inttostr(round((ScrollBar7.Position)/(446-42)*135-13));
2406:      Timer6.Enabled:=false;
2407: end;
2408:
2409: procedure TForm1.PageControl1Change(Sender: TObject);
2410: begin
2411:      if PageControl1.ActivePage=TabSheet2 then
2412:         Timer6.Enabled:=true;
2413:      if PageControl1.ActivePage=TabSheet4 then begin
2414:         Timer4.Enabled:=true;
2415:         Timer5.Enabled:=true;
2416:      end;
2417: end;
2418:
2419: procedure TForm1.SpeedButton17Click(Sender: TObject);
2420: begin
2421:      Timer4.Enabled:=true;
2422:      Timer5.Enabled:=true;
2423: end;
2424:
2425: end.
```

Bibliografía

Introducción

[1] A. Guyton. Tratado de Fisiología Médica.
Editorial Interamericana - McGraw Hill. 2001

[2] Merill E W. Rheology of Blood;
Physical. Review. Vol.49 1985; pp. 863–888

[3] D. Durrer, et al, Total Excitation of the Isolated Human Heart.
Circulation, Vol. 41, 1985; pp. 899 –912

[4] W. Hofmann. Biophysics.
Editorial Mc Graw Hill. 2003

[5] José M. Ferrero Corral. Bioelectrónica, Señales Bioelectrónicas.
Editorial : Servicio de Publicaciones de la Universidad Politécnica
de Valencia. 2002

[6] Easthope and Brooks. A Comparison of Rheological
Constitutive Functions for Whole Human Blood;
Biorheology Vol.17,1995; pp. 235 -247

[7] Antonio Cresus. Simulación y Control de Procesos por Ordenadores.
Editorial Marcombo. 2001

[8] Naylor Thomas. Experimentos de Simulación en Computadoras con
Modelos de Sistemas Económicos.
Editorial Limusa. 1997

[9] Gordon Geoffrey. Simulación de Sistemas.
Editorial Mc Graw Hill. 2003

[10] Ferziger, J.H. Numerical Methods for Engineering Application.
Editorial Wiley, New York. 1981

[11] Kamlesh Mathur, Daniel Solow. Investigación de Operaciones.
Editorial Prentice-Hall Hispanoamericana SA. 1996

Capítulo 1

[12] P. Roach. Computacional Fluid Dynamics.
Editorial Hermosa Publishing. 1972

[13] J.Connor and C. Brebbia. Finite Element Techniques for Fluid flow.
Editorial Newnes-Butterworths. 1976

[14] J.Dupuy and A.Dianoux. Microscopic Structure and Dynamics of Liquids.
Editorial Plenum Press. 1978

[15] J. Goodfellow. Molecular Dynamics.
Editorial Macmillan Press. 1991

[16] P.L. Bhatnagar, E.P. Gross, and M. Krook. A model for collisión processes in gases.
Physical. Review. Vol.94 1954; pp. 511–525

[17] Sauro Succi. The Lattice Boltzmann Equation.
Editorial : Oxford University Press, New York. 2001

[18] G. McNamara and G. Zanetti. Use of the Boltzmann Ecuation to Simulate Lattice Gas Automata.
Physical. Review. Vol.61 1988; pp. 233

[19] S.Chen, H.Chen, D.Martínez and W.Mattaeus. Lattice Boltzmann Model for Simulation of Magnetohydrodynamics.
Phys. Rev. Letts. 67(27), 3776, 1991

[20] P.Bhatnagar, E. Gross and M. Krook. A Model for Collision Process in gases I: Small Amplitude Processes in Changed and Neutral One-Component System.
Phys. Rev. 94, 511, 1954.

[21] T.Abe. Derivation of the Lattice Boltzmann Method by Means of the Discrete Ordinate Method for the Boltzmann Equation.
J. Comp. Phys. 131, 241, 1997.

[22] J. M. Koelman. A Simple Lattice Boltzmann Scheme for Navier Stokes Equations.
Europhysics Letters, 15, 1991.

[23] X.He and L.S.Luo. A Priori Derivation of the Lattice Boltzmann Equation.
Phys. Rev. E55, 6333, 1997.

[24] X.Shan and X.He. Discretization of the Velocity Space in the Solution of the Boltzmann Equation.
Phys. Rev. Lett. 80(1), 65, 1998.

[25] B.Elton, D.Levermore and G.Rodriguez. Convergence of Convective Diffusive Lattice Boltzmann Methods.
SIAM J. Num. Anal. 32(5), 1327, 1995

[26] Z. Guo, B. Shi y C. Zheng. A Coupled Lattice BGK Model for the Boussines Equations.
International Journal for Numerical Methods in Fluids, 39, 2002

[27] B. J. Palmer y D. R. Rector. Lattice Boltzmann Algorithm for Simulating Thermal in Compressible Uids.
Journal of Computational Physics, 161, 2000

[28] Y. Qian. Simulating Thermohydrodynamics with Lattice BGK Models.
Journal of Science and Computation, 8, 1993

[29] Y. Chen, H. Ohashi y M. Akiyama. Thermal Lattice Bhatnagar-Groos-Krook Model Without Nonlinear Deviations in Macrodynamic Equations.
Physics Review E, 50, 1994

[30] Einat Aharonov and Daniel H. Rothman. Non-Newtonian Flow (Through Porus media) : A Lattice Boltzmann Method.
Physical. Review. Vol.43 1993; pp. 680

[31] N. Rakotomalala, D. Salin and P. Watzky. Simulation of Viscous Flow of Complex Fluid with a Bhatnagar, Gross and Krook Lattice Gas.
Phys. Fluids Vol.8 1996; pp. 320

Capítulo 2

[32] Q. Zou and X. He. On Pressure and Velocity Flow Boundary Conditions for the Lattice Boltzmann BGK Model.
Phys. Fluids Vol.9 1997; pp. 1591

[33] Q. Zou and X. He. Analysis and Boundary Condition of the Lattice Boltzmann BGK Model with Two Velocity Components.
Los Alamos preprint, LA-UR-95-2293

[34] X. He, L.S. Lue and M.Dembo. The Dirichlet Boundary Condition in Hydrodynamics and Analytic Solutions of Simple Flows for the Lattice BGK Method.
Phys. Fluids Vol.11 1994; pp. 2201

[35] D.R.Noble, S.Chen, J.G.Georgiadis, R.O.Buckius. A Consistent Hydrodynamic Boundary Condition for the Lattice boltzmann method.
Phys. Fluids Vol. 7 1995; pp. 203

[36] R.S.Maier, R.S.Bernard and D.W.Grunau. Boundary Conditions for the Lattice Boltzmann Method.
Preprint (1995)

[37] Yue-Hong Qian and Ye Zhou. Complete Galilean-Invariant Lattice BGK Models for the Navier-Stokes Equation.
Phys. Rev. 1998

Capítulo 3 y 4

[38] Sugii Y. Nishio S. and Okamoto K. Measurement of a Velocity Field in Microvessels Using a High Resolution PIV Technique.
Ann. N.Y. Acad. Sci. 972: 331-336. 2002

[39] Intaglietta M, Thompkins WR, and Richardson DR. Velocity Measurements in Microvasculature of the Cat Omentum by on-line Method.
Microvasc Res, 2: 151-162,1970.

[40] Bishop JJ, Nance PR and Intaglietta M. Effect of Erythrocyte Aggregation on Velocity Profiles in Venules.
Am. J Physiol Heart Circ Physiol, 280: H222-H234, 2001

[41] Bishop JJ, Johnson PC and Intaglietta M. Effect of Erythrocyte Aggregation and Venous Network Geometry on Red Blood Cell Axial Migration.
Am. J Physiol Heart Circ Physiol, 281: H939-H950, 2001

[42] C.Migliorini, Y.Quian, H.Chen and E.Brown. Red Blood Cells Augment Leukocyte Rolling in a Virtual Blood Vessel.
Biophysical Journal Vol.83 2002; pp. 1834-1841

[43] Cabrales P. Amy G. Tsai and Intaglietta M. Microvascular Pressure and Functional Capillary Density in Extreme Hemodilution with Low and High Viscosity Dextran and a Low Viscosity Hb-Based O_2 Carrier.
Am. J Physiol Heart Circ Physiol, 280: H232-H532, 2004

[44] Bishop JJ, Nance PR and Intaglietta M. Erythrocyte Margination and Sedimentation in Skeletal Muscle venules.
Am. J Physiol Heart Circ Physiol, 280: H222-H236, 2001

[45] M.A. Hussain, Subir Kar and Ram R. Puniyani. Relationship Between Power Law Coefficients and Major Blood Constituents Effecting the Whole Blood Viscosity.
Journal of Biosciences 24(3): 329-337 sep 1999

[46] Toledo, R.T. Fundamentals of Food Process Engineering. 2nd Edition.
Editorial Prentice Hall. 1991

[47] Steven C.Chapra y Raymound P.canale. Métodos Numéricos para Ingenieros, 4ª edición.
Editorial Mc Graw Hill. 2004

[48] Kristopher Cunningham and Avrum Gotlieb. The Role of Shear Stress in the Pathogenesis of Atherosclerosis.
Laboratory Investigation (2005) 85, 9-23.

[49] Carlos E. Ciancaglini. Hidrodinámica de la Circulación Vascular Periférica Normal y Patológica.
Cardiol (2003) 32: 259-279.

[50] M.Krafczyk, M.Cerrolaza and E.Rank. Analysis of 3D Transient Blood Flow Passing Through an Artificial Aortic Valve by Lattice-Boltzmann Methods.
Journal of Biomechanics 31(3): 453-462 oct 1997