# **Apuntes – Módulo 5**

## **Curso: Modeling and Simulation of Natural Processes**

**Universidad de Ginebra - Prof. Bastien Chopard et al.** **Semana 5 – Modelado de Flujo de Fluidos con Lattice Boltzmann**

**1. Dinámica de Fluidos Computacional (CFD): Una Vista General**

Esto sirve bastante para entender el panorama general y por qué es tan importante este campo de aplicaciones.

* **¿Qué es un fluido?**

Básicamente, cualquier sustancia que se deforma continuamente bajo tensión de cizallamiento. O sea, líquidos, gases, plasmas… lo contrario a un sólido que resiste esas deformaciones. Se modelan como un "continuo", no como partículas individuales

* **¿Por qué estudiar fluidos?**
  + Para entender cómo los objetos interactúan con el medio donde están sumergidos
  + Las ecuaciones que describen los fluidos (Navier-Stokes) son súper difíciles de resolver analíticamente, así que la simulación numérica es nuestra mejor amiga.
  + Las aplicaciones son infinitas: desde la física y la química hasta la biología (flujo sanguíneo), la geología, etc.
* **Ejemplos que nos mostró Jonas:**
  + **Convección de calor:** Cómo el aire acondicionado distribuye el aire en una sala. Se ve cómo cambia el patrón de flujo si el aire acondicionado es fijo o si "barre" la sala. Esto es vital para la eficiencia energética.
  + **Flujo sanguíneo en una arteria:** Impresionante ver cómo modelan la interacción entre la sangre (fluido) y la pared de la arteria (sólido deformable). Aquí el acoplamiento bidireccional es clave.
  + **Lavadora:** Hasta para entender cómo el agua interactúa con la ropa es útil.
  + **Colapso de una columna de agua:** Visualmente muy impactante.
* **Valor de la CFD:** Podemos obtener perspectivas que son imposibles de lograr con experimentos físicos (ej., visualizar el campo de velocidades en cada punto). Esto es una gran ventaja para la investigación y el diseño.

### **2. Ecuaciones fundamentales y retos en CFD**

Aquí entramos en la teoría y las matemáticas detrás.

* **Ecuaciones de Navier-Stokes (ENS):** Son la base. Hoy nos centramos en la versión para flujo incompresible.
  + **Primera línea:** Conservación del momento. ¡Son en realidad 3 ecuaciones para las 3 componentes del momento!
  + **Segunda línea:** Conservación de la masa (o la incompresibilidad del flujo, ∇⋅u=0).
  + **Consideración:** Son "simples" porque no tienen en cuenta efectos como compresibilidad, temperatura, reacciones químicas, etc., pero aun así son súper útiles para líquidos y muchos gases.
* **Retos en la resolución de las ENS:**
  + Son no lineales.
  + Están acopladas (las variables de velocidad y presión están interconectadas).
  + Son Ecuaciones Diferenciales Parciales (EDP): dependientes del espacio y del tiempo.
  + Implican diferentes escalas de longitud y tiempo, lo que las hace costosas computacionalmente.
* **Condiciones de contorno (BCs):** ¡Tan importantes como las ecuaciones mismas!
  + Físicas: Alrededor de obstáculos (ej., un cilindro). Aquí el fluido debe respetar las propiedades de la superficie del objeto.
  + Virtuales: En los límites del dominio de simulación (entrada, salida, arriba, abajo). No tienen un significado físico directo, pero son cruciales para simular un dominio más grande y minimizar las perturbaciones por truncamiento. La idea es "fingir" que el sistema es más grande.

### **3. Del Gas Reticular (LGA) al Lattice Boltzmann (LBM)**

Este es el puente entre lo que vimos en el Módulo 4 y lo de ahora.

* **Recordatorio LGA (HPP):** Intentaba imitar la física real de un gas con moléculas idealizadas en una rejilla discreta (máx. 4 partículas/celda, apátridas). Tenía dos pasos:
  + **Colisión:** Partículas en la misma celda interactúan y reorientan sus direcciones.
  + **Propagación (Streaming):** Partículas se mueven a celdas vecinas.
* **Problemas con LGA:**
  + **Ruido estadístico:** Como trabaja con números enteros de partículas, hay mucho ruido. Necesitas muchísimas partículas para que se parezca a un fluido real.
  + **Anisotropía:** Las redes cuadradas (HPP) no tienen suficiente simetría para modelar adecuadamente ciertos flujos.
  + **Invariantes espurios:** Cosas como la conservación de momento por filas/columnas o el "efecto tablero de damas" (partículas en celdas blancas solo van a negras y viceversa, no interactuando con todas las demás). Esto no es físico.
* **Ahora entra Lattice Boltzmann (LBM)**
  + Abandona la idea de "partículas" individuales y trabaja con "poblaciones" (fi​).
  + Estas poblaciones fi​ se interpretan como densidades de probabilidad de encontrar moléculas con una velocidad discreta específica vi​ en un punto y tiempo dados. Esto viene de la Teoría Cinética de Gases y la Ecuación de Boltzmann.
  + La evolución de estas poblaciones también tiene dos pasos:
    1. **Colisión:** Instantánea, local en la celda. Las poblaciones fi,in​ (antes de colisión, verdes) se transforman en fi,out​ (después de colisión, rojas).
    2. **Propagación (Streaming):** Las fi,out​ de una celda se convierten en fi,in​ de las celdas vecinas. ¡Aquí es donde avanza el tiempo!
  + **Discretización:** Tenemos Δt (paso de tiempo) y Δx (distancia entre celdas).
* **Ventaja clave del LBM:** Mucho más eficiente que LGA, menos ruido.

### **4. Variables macroscópicas**

Esto es cómo conectamos nuestras "poblaciones" discretas con el mundo real de los fluidos

* **Densidad (ρ)**: La suma de todas las poblaciones en una celda.
  + ρ=∑i​fi​
* **Velocidad (u)**: Se calcula a partir del momento.
  + ρu=∑i​vi​fi​
  + Es decir, sumamos el producto de cada velocidad discreta por su población correspondiente, y luego dividimos por la densidad.
* **Poblaciones vs. Velocidades Reticulares:**
  + **Poblaciones (fi​)**: Son *escalares* que cambian en espacio y tiempo. Se dibujan como flechas solo para recordar su dirección de propagación. Su valor (magnitud) se visualiza por el "grosor" de la flecha.
  + **Velocidades Reticulares (vi​)**: Son *vectores constantes* (las 9 direcciones posibles). Son fijos y no cambian.
* Hay que tener claro qué significa cada flecha o símbolo para no confundirse.

### **5. El Paso de colisión: El modelo BGK**

Aquí es donde las poblaciones "interactúan" para llevar al sistema al equilibrio.

* **Interpretación de poblaciones:** Ya no son partículas, sino promedios de moléculas. Se rigen por la Ecuación de Boltzmann, de ahí el nombre del método.
* **Objetivo de la colisión:** Llevar las poblaciones al **equilibrio**.
* **Distribución de equilibrio (fieq​):** Es una función conocida (usualmente una expansión de Taylor de la Maxwell-Boltzmann) que depende de las variables macroscópicas (ρ y u) locales.
  + La fórmula incluye términos constantes, lineales y cuadráticos en la velocidad. (Me anoto que tengo que tener cuidado con los productos escalares y los cuadrados de estos en la implementación).
  + Es importante que la fieq​ cumpla las leyes de conservación de masa y momento.
* **Modelo BGK (Bhatnagar-Gross-Krook):** Es una simplificación del término de colisión de la ecuación de Boltzmann.
  + La idea es que las poblaciones se relajan hacia su estado de equilibrio a una cierta tasa.
  + Fórmula: fiout​=fiin​−ω(fiin​−fieq​)
  + ω es el parámetro de relajación. Es crucial porque controla la viscosidad del fluido y la velocidad a la que el sistema alcanza el equilibrio. Si ω es grande, el sistema se relaja rápido.
  + Impresionante que con NumPy, la colisión se resuelva en una sola línea de código

### **6. El Paso de propagación (Streaming)**

Este es el paso donde las poblaciones "se mueven" a través de la rejilla.

* **¿Qué hace?**

Mueve las poblaciones *post-colisión* (fi,out​) de una celda a las *pre-colisión* (fi,in​) de las celdas vecinas, según sus direcciones vi​. Es decir, los valores fi,out​ de (x, y) en el tiempo t se convierten en fi,in​ de (x + v\_i.x, y + v\_i.y) en el tiempo t + Δt.

* **Implementación en Python:**
  + El desafío está en manejar los límites del dominio.
  + Condiciones de contorno periódicas: Lo más fácil para empezar. Lo que sale por un lado, entra por el opuesto (izquierda/derecha, arriba/abajo). Esto es muy útil para pruebas iniciales, pero no siempre realista.
  + Truco de "swap" (intercambio): Para trabajar con un solo conjunto de poblaciones (sin f\_in y f\_out separados), se puede hacer un intercambio directo de valores entre celdas vecinas.

### **7. Condiciones de contorno (Aplicación práctica)**

Vuelven las BCs, pero ahora con detalles de implementación en LBM.

* **Importancia:** Son tan vitales como las ecuaciones y difíciles de implementar correctamente.
* **Tipos de BCs:**
  + **Físicas (ej., Obstáculo):** En la superficie del obstáculo, no hay fluido dentro. La condición de "no deslizamiento" (velocidad del fluido es cero en la superficie) se implementa con una "regla de rebote".
    - **Regla de Rebote:** Una partícula que llega a la superficie con una dirección i (ej., f1​ que va a la derecha), "rebota" y se convierte en una partícula que sale en la dirección opuesta i+4 (ej., f5​ que va a la izquierda). Básicamente, fi​→fi+4​. Esto asegura que el flujo perpendicular sea cero y la velocidad tangencial sea nula.
  + **No físicas (Virtuales):**
    - **Condición de Entrada (Inflow):** Necesitamos imponer un perfil de velocidad (ej., flujo de Poiseuille parabólico para tuberías, o un flujo constante en la entrada de un dominio).
      * Desafío: Tenemos fi​ que *entran* al dominio (conocidos) y fi​ que *salen* (desconocidos).
      * Estrategia: Las poblaciones que salen se calculan asumiendo un equilibrio local, mientras que las que entran se calculan para imponer la velocidad deseada. Es un poco más complejo, involucra el cálculo de la densidad primero y luego usar la feq​ para las poblaciones que entran.
    - **Condición de Salida (Outflow):** El dominio no debe terminar bruscamente. Queremos que el fluido "fluya" sin perturbaciones.
      * Estrategia simple: Simplemente copiamos las poblaciones de la celda *anterior* (dentro del dominio) a las poblaciones desconocidas en la frontera de salida. Esto simula una condición de "gradiente cero" para esas poblaciones.

### **8. Flujo alrededor de un obstáculo (Aplicación)**

La aplicación real de todo lo que se ha visto.

* **Configuración:** Un sistema 2D con un cilindro (obstáculo) en el medio. Flujo de fluido desde la izquierda.
* **Código:** Todo el código es muy compacto (unas 100 líneas en Python con NumPy). Esto demuestra la eficiencia del LBM.
* **Resultados y Comportamiento:**
  + **Número de Reynolds bajo (ej., Re = 1):** El flujo es estable. Después de un transitorio inicial, converge a un estado estacionario donde las velocidades no cambian. Se ve el flujo simétrico alrededor del cilindro.
  + **Número de Reynolds más alto (ej., Re = 220):** Aquí la cosa se pone interesante. Tras un transitorio, el flujo se separa detrás del obstáculo y se vuelve inestable. Se rompe la simetría y se empiezan a ver oscilaciones en la estela. Esto es el famoso sendero de vórtices de von Kármán
    - **Consideración personal:** Es increíble que un modelo tan "simple" como Lattice Boltzmann pueda capturar un fenómeno físico tan complejo y observable en la naturaleza (como el sonido de los cables de teléfono por el viento) Esto realmente muestra el poder de estas simulaciones.
* **Capacidad del código:** Podemos cambiar la geometría, los parámetros (velocidad, viscosidad, etc.) y explorar nuevos comportamientos.

**Consideraciones generales del módulo:**

Este módulo ha sido denso pero muy completo. Pasar de la teoría abstracta a ver cómo se implementa un modelo de Lattice Boltzmann en Python y cómo reproduce fenómenos físicos como el sendero de von Kármán es una representación muy buena del potencial de aplicación en ingeniería. Me queda claro que el LBM es una herramienta muy potente y eficiente para la CFD, y se da a entender la importancia crítica de las condiciones de contorno.