TECNOLÓGICO NACIONAL DE MÉXICO

Instituto tecnológico de Celaya

Departamento de ciencias básicas

Ecuaciones diferenciales

Simulación de vórtices de vórtices Von Karman

Ernesto Ramirez Hernandez

30/05/2022

Los tipos de flujo

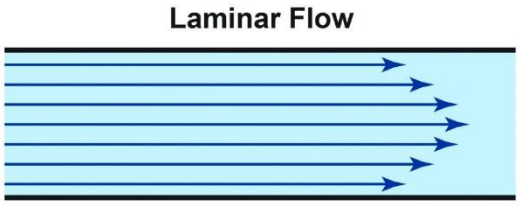
"*La prueba de la ciencia es predecir ¿Alguna vez has visitado la tierra? ¿Podrías predecir las tormentas, los volcanes, las olas de los océanos, las auroras, y el atardecer?*

-Richard Feynmann (Segundas lecturas)

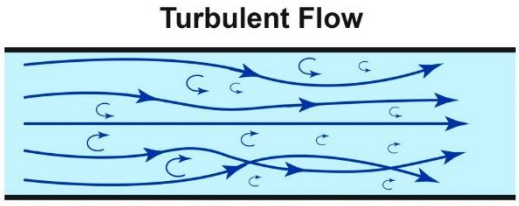
La turbulencia es un efecto de la naturaleza que siempre nos ha llamado la atención, a pesar de que se ha tratado de manipular los fluidos para hacerlos más "Fáciles" de comprender la realidad es que por más que nos esforzamos el flujo turbulento siempre aparece, desde el interior de tu cuerpo hasta las alas de un avión y a pesar de que artificialmente creamos flujos que son controlables, la realidad es que nunca se pueden controlar completamente.

Flujo Laminar y flujo turbulento

Dentro de la física con fluidos newtonianos existen dos tipos de flujo cuando sus partículas se mueven: Flujo laminar y flujo turbulento.



El flujo laminar aparece cuando el fluido se desplaza en un mismo volumen y en una misma dirección a través de ambiente controlado y se caracteriza por tener un flujo paralelo de partículas que viajan a una velocidad uniforme.



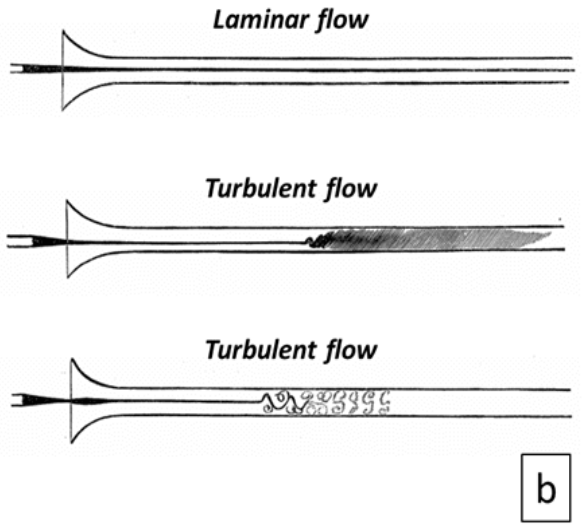
El flujo turbulento es todo lo contrario al flujo laminar, el flujo turbulento aparece cuando las partículas paralelas del flujo laminar desaparecen y empiezan a viajar de manera completamente desorganizada y se pierde ese flujo controlado del fluido.

Características del flujo turbulento

El flujo turbulento es mas que simple desorden, realmente lo que en gran parte crea un flujo turbulento son una cantidad enorme de torbellinos (También llamados vórtices) esa es una de sus características, otra es que a diferencia del flujo laminar el flujo turbulento aparece entre más rápido viaje el fluido. Esto lo descubrió el matemático Osbourne Reynolds, estaba experimentando con un flujo de tinta que pasaba en un cilindro, cuando el fluido viajaba a una velocidad lenta se mantenía el flujo, pero conforme aumentaba la velocidad el fluido empezaba a mostrar un cierto descontrol y cuando lo llevaba más allá del punto critico se dio cuenta que la tinta se mezclaba con el agua, aquí Reynolds descubre

otra propiedad del flujo turbulente, y es que es difusivo, por lo que si dos fluidos laminares alcanzan la velocidad suficiente, empezaran a volverse turbulentos lo que causara que se mezclen, esto hace que los fluidos aumenten no solo de tamaño sino que también afecta el momento del fluido e incluso la temperatura, cambios que se distribuyen por todo el fluido.

Pero Reynolds no solo descubrió esto, también se dio cuenta de que el flujo cambiaba dependiendo de donde se encontraba el fluido



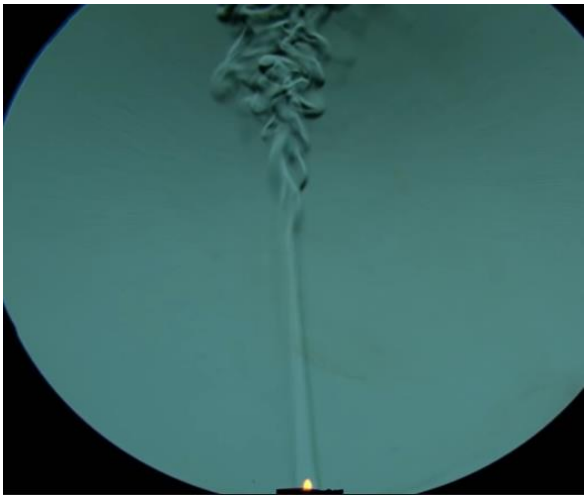
De esta manera aparecio el llamado numero de Reynolds, este numero dicta que tan probable sea que tu flujo sea turbulento tomando en cuentra tres variables: La velocidad a la que esta viajando un fluido, la longituid caracteristica del ambiente (En este caso el ancho de la tuberia) y la viscosidad cinemtaica que se puede pensar como la medida de su friccion interna.

$$Re_{Adimensional} = \frac{u \cdot L}{v}, donde: u = Velocidad\ del\ flujo,$$

$$L = Longitud\ caracteristica,$$

$$v = viscosidad\ cinematica$$

Un mayor número de Reynolds significa que el fluido esta viajando rápido, es poco viscoso y se encuentra en un ambiente espaciosos y esto es una propiedad de un fluido turbulento, un numero de Reynolds alto.



Esto se puede ver con la llama de una vela (Foto a la izquierda) conforme la estela del humo va subiendo también su velocidad y por lo tanto también el tamaño del ambiente que la rodea, por eso conforme sube el flujo se vuelve turbulento por el aumento de la velocidad del número de Reynolds.

Casi todos los fluidos son turbulentos, hay estudios que muestran que el flujo turbulento es esencial para la formación de gotas de lluvia.

Otra característica de la turbulencia es la capacidad disipativa, esto significa que conforme una fuerza se aplica al flujo, este tiende a disipar esa energía a través de sus vórtices, los cuales van disminuyendo su tamaño hasta que convierten esa energía en calor, por eso se suelen pensar en los flujos turbulentos cuando tienes estructuras que alteran el movimiento de un fluido como los aviones.

La última característica, pero no menos importante es que el flujo turbulento es impredecible, no se tiene un modelo preciso para calcular en qué dirección se moverá un fluido que se mueve de forma turbulenta, pero eso no ha impedido que nazcan modelos que traten de explicar como se disipan estas fuerzas, por lo tanto, han nacido varios modelos gracias a una formula en específico.

Ecuación de Navier Stokes

En física, las ecuaciones de Navier-Stokes son un conjunto de ecuaciones diferenciales parciales no lineales que describen la evolución temporal y espacial del momentum y energía de un fluido turbulento, nombradas en honor al ingeniero y físico francés Claude-Louis Navier y al físico angloirlandés George Gabriel Stokes.

$$\nabla \cdot u = 0$$

Esta primera función representa el factor de divergencia, que marca que en un campo vectorial de la velocidad del fluido debe ser igual a cero.

$$\frac{\partial u}{\partial t} + (\nabla \cdot u)\, u = -\frac{1}{\rho}\nabla p + \mu \nabla^2 u + F$$

En donde la igualdad izquierda representa la aceleración del fluido y la derecha la fuerza.

Donde:

u: $velocidad$ $(Campo\ vectorial\ en\ dos\ dimensiones)$

$p = Presion$

$F = Fuerza$ $(representa\ \rho \cdot g)$

$\mu = Viscosidad\ cinematica$

$\rho = Densidad$

$t = tiempo$

$\nabla =$ Operador Nabla (Definiendo conveccion no lineal, gradiente y divergencia)

$\nabla^2 = Operador\ de\ Laplace$

Antes de explicar esta ecuación tenemos que dejar en claro que es una ecuación irresoluble,

por lo que se necesita hacer una derivación de la ecuación, con ciertas restricciones claras:

1. Que el fluido sea newtoniano: Esto significa que la densidad del fluido no cambia

dependiendo de la aceleración y la fuerza que se le aplique.

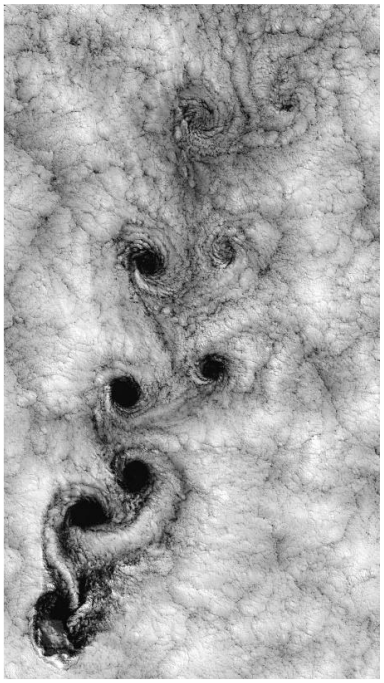
2. El fluido debe ser incompresible: El fluido a pesar de aplicársele una presión

superficial o total, esto no causara cambios considerables en el volumen del fluido.

3. Tener propiedades isotérmicas: Esto significa que conforme el fluido se vuelve

turbulento no hay perdidas considerables de calor.

Esta es una demostración simple de como la ecuación de navier-stokes a pesar de tener reputación de ser irresoluble, nos muestra su utilidad ¿Pero que pasa cuando tienes que modelar un fluido turbulento en cantidades grandes? Y no solo eso, sino también graficar sus efectos de la turbulencia.



En este caso los Vórtices de Von Karman, estos vórtices aparecen generalmente en los fluidos turbulentos a ser parte esencial de los mismos, la cual es torbellinos en varias dimensiones en lo ancho y alto del fluido, pero a pesar de esto los vórtices de Von Karman, son uno de los pocos fenómenos que aparecen en los fluidos turbulentos y que parecen tener un patrón replicable, pero difícil de calcular a grande escala a mano, para esto aparecen los simuladores informáticos, en donde nos topamos con otro problema, y es que la ecuación de navier-stokes a pesar de ser una ecuación que esta destinada a dominar todos los fluidos turbulentos la realidad es que es una ecuación que toma demasiado tiempo de resolver incluso con métodos de diferenciación exactos, sin embargo puede ser útil para hallar otros métodos que nos ayuden a modelar estos modelos para poder estudiarlos.

Método Enrejado de Boltzmann

El método de Enrejado de Boltzmann es un método poderoso que se ha hecho famoso por es un método eficiente capaz de competir con los modelos clásicos de simulación de fluidos como el de elementos finitos o el de métodos de volumen infinitos.

Pero para pode comprender a fondo cómo funciona este método se deben explicar varios conceptos físicos para poder llevar una simulación correcta.

1. Diferencias de la escala microscópica a macroscópica
   Los fluidos a escala humana se ven muy distintos una vez que los ves a escala microscópica, por lo que nosotros cuando vemos un fluido en reposo, en realidad a escala microscópica el fluido se sigue moviendo y no precisamente poco.
   La mayoría de los programas de simulación de fluidos son solucionadores de que integran numéricamente la ecuación de Navier Stokes usando diferencias finitas o métodos que usan metodologías con iteraciones finitas.

$$\frac{\partial u}{\partial t} + (\nabla \cdot u)\, u = -\frac{1}{\rho}\nabla p + \mu \nabla^2 u + F$$

   A pesar de ser una ecuación compleja se puede resolver en una estructura lagrangiana o euleriana, dependiendo del problema en estudio.

   Esta ecuación la densidad y velocidad son continuas a escalas macroscópicas por lo que implica que:

   - Se pueden describir como funciones continuas
   - Las podemos medir con instrumentos a escala humana

   Debemos tomar en cuenta que la forma básica de medir la densidad puede ser con una balanza, pero a nivel microscópico esto se vuelve mas complicado, porque como estamos midiendo la densidad de partículas en un área determinada esto significa que tenemos que hacer un conteo de estas.

   A pesar de que es posible describir coherentemente un fluido a escala macroscópica la realidad es que, en una escala microscópica, se entra a una dimensión en donde las partículas pueden comportarse de manera completamente diferente.

2. Enlace entre escalas microscópicas y macroscópicas
   Por un momento podríamos pensar que las bases microscópicas de los fluidos son irrelevantes para el movimiento de los fluidos a un nivel de percepción humano, pero realmente nos ayuda a comprender mejor la mecánica de los mismos.

El físico Ludwig Boltzmann (1844-1904) se figuró tres preguntas:

- ¿Como puedes definir la densidad de un fluido? Siendo el mismo un cumulo de partículas en movimiento.
- ¿Cómo podrías definir la velocidad de un fluido?
- ¿Cuál es el vinculo entre el movimiento de partículas y las propiedades observables de un fluido?

Estas preguntas fueron las que ayudaron a Boltzmann a sentar las bases para este método.

3. Caso de estudio de un gas en reposo a una velocidad cero

   Si observamos un cuarto que esta lleno de un gas, en este caso aire, no podríamos observar nada a nivel macroscópico, pero a nivel microscópico las partículas se están moviendo por lo que realmente significa la densidad es la suma de el número de partículas.

$$\rho = \frac{N_p + m_p}{V}$$

$$m_p = Masa\ de\ la\ particula$$

$$N_p = Numero\ de\ particulas$$

$$V = Volumen\ del\ cuarto$$

   Es importante recalcar que aquí no se toman en cuenta lo choques entre partículas ni sus subsecuentes vibraciones.

4. Isotropía

   La isotropía representa la probabilidad de que la velocidad hacia una cierta dirección es la misma para todas las partículas, por lo que a nivel macroscópico no representa algo muy importante, pero después veremos como a nivel de programación esto resulta esencial para hacer la simulación.

   En esta parte se hace la llamada “suposición de la isotropía” explica que como estamos observando en escala humana haremos caso omiso por el momento acerca de las perturbaciones que tengas las partículas las unas con las otras, por lo tanto, la fuerza cinemática debe ser diferente a cero:

$$\frac{\|v_1\| + \|v_2\| + \|v_3\|}{3} \neq 0$$

   La suma de los vectores debe de ser diferente a cero.

5. Posición y velocidad de partícula - Distribución de Maxwell

   La distribución de Maxwell nos muestra que dependiendo de la temperatura del fluido este se va a comportar de manera diferente, pero como mencionamos al principio que íbamos a trabajar con un fluido isotérmico, la temperatura no se tomara en cuenta. Sin embargo, hay que mencionar que conforme la temperatura cambia así mismo la velocidad cinética promedio se verá afectada.

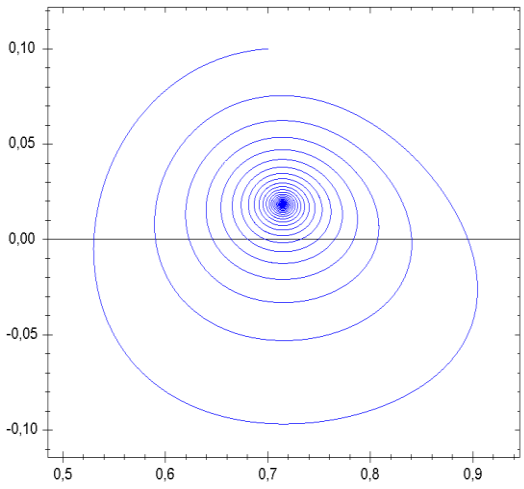
   Asumiendo que la partícula puede tener una velocidad con un valor promedio, no nos da una idea clara de como se distribuyen las velocidades.

   Si consideramos que cada partícula individual posee una velocidad de acuerdo con una ley de probabilidad indeterminada, el teorema de limite central nos enseña que, para una cantidad grande de partículas, la ley de distribución puede ser aplicada a la Ley de normalidad Gaussiana por lo que, la distribución de las magnitudes de la velocidad se puede hacer de acuerdo con la ley normal que dice:

$$f^{eq}(\rho, u) = \rho(2\pi R_s T)^{-1} \exp\left((e-u)^2 (2R_s T)^{-1}\right)$$

$$donde\ R_s = es\ la\ constante\ especifica\ del\ gas$$
$$T = temperatura$$
$$e = velocidad\ microscopica$$
$$u = velocidad\ macroscopica$$

6. Explicación del espacio fásico



Generalmente en nuestra mente cuando queremos graficar una función, solemos tratar de poner la función en un sistema de coordenadas cartesianas, pero en este caso como el movimiento está ligado al tiempo se tiene que generar un espacio fásico, este espacio es una construcción matemática que permite representar el conjunto de posiciones y momentos conjugados de un sistema de partículas. En grandes rasgos el espacio fásico se corresponde a la recolección de geometría y espacios de velocidad.

Esta distribución de partículas puede ser definido como una función $f\,(x, e, t)$ que corresponde a la a la probabilidad de que una partícula en una posición [x] posea la velocidad [e] en el momento [t].

Siguiendo el principio de inercia, sabemos que la distribución de la velocidad se preserva conforme pasa el tiempo, en ausencia de un campo de fuerzas exteriores y en la ausencia de colisiones entre partículas.

$$f\left(x+\delta x, e+\frac{\delta x}{\delta t}, t+\delta t\right)-f(x,e,t)=0\}$$

La transformada de la expresión f puede ser obtenida con un desarrollo de Taylor:

$$Df=f+\delta_x\partial_x f+\frac{\delta_x}{\delta_t}\,\partial_e f+\delta_t\partial_x f-f=0$$

Ahora bien, si eliminamos las fuerzas externas:

$$\partial_x f+e\nabla_x f=0$$

En el caso de que el fluido tenga colisiones esta igualdad debe de ser modificada tomando en cuenta las variaciones de la probabilidad de distribución en el espacio fásico.
A este factor se la va a llamar operador de colisión:

$$\partial_x f+e\nabla_x f=(\partial_t+v\nabla)$$

De esta manera hemos reescrito la ecuación del Enrejado de Boltzmann, el operador de colisión puede ser la cantidad de variación, correspondiente hacia un movimiento que tiende al equilibrio de distribución, por lo tanto, se puede hacer una ecuación similar a la fórmula de distribución de maxwell.

$$(\partial_t+v\nabla)=-\left(\frac{f^{eq}-f}{\tau}\right)$$

A esta ecuación se le suele llamara aproximación BGK.

Una vez explicados todos los conceptos previos, ahora si podemos hacer una codificación para hacer la simulación de un fluido chocando con un cilindro, de esta manera se podrán observar los vórtices de Karmann una vez que el fluido llega a su capa limite.

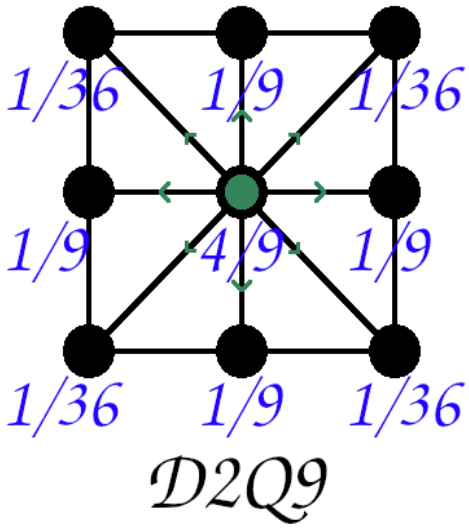
En resumen

- Para un fluido en reposo, la suma vectorial de las velocidades es nula.
- En escala macroscópica cada partícula tiene una velocidad que no es nula
- La velocidad de las partículas puede ser representada con la fórmula de equilibrio de maxwell

Implementación del método de Enrejado de Boltzmann

Como mencionamos antes, el método de Enrejado de Boltzmann (Al cual llamaremos EB) es mas eficiente que cualquier método que use directamente las ecuaciones de Navier-Stokes, esto pasa porque en vez de hacer la evolución con la fórmula de Stokes, que como observamos en un fluido simple, es una ecuación muy complicada de usar.

El método EB funciona gracias a una vasta red que tiene nodos, cada nodo tiene 9 posibilidades distintas de desplazar una partícula, y cada partícula se encuentra en un nodo, esto reduce enormemente la alta dimensionalidad de las propiedades microscópicas del fluido en una red que tiene límites de desplazamiento claros.



Este sistema se le llama D2Q9, porque la formula original es DmQn donde m representa las dimensiones y n la cantidad de redes que los une, teóricamente para tres dimensiones será D3Q27, pero en este caso por las restricciones que mencionamos con la ecuación de Stokes nos limitaremos a hacerlo de dos dimensiones.

Cada nodo tiene un numero especifico el cual es su peso representado por W_i las partículas microscópicas que conforman al fluido se pueden describir como una función $f(x, v)$, la cual describe la densidad del fluido en el espacio físico en la localización x viajando a la velocidad v.

Las partículas tendrán dos comportamientos: Fluir y colisionar, este comportamiento ya lo conocemos por la aproximación BGK que ya vimos antes.

$$(\partial_t + v\nabla) = -\left(\frac{f^{eq} - f}{\tau}\right)$$

Por lo que podemos empezar a codificar.

Se usará el lenguaje Python (versión 3.9) con el entorno de depuración Anaconda, popularmente usado para matemática en Python.

Se usarán también dos librerías:

- Numpy: Para hacer más rápido ciertas operaciones como potencias y sumatorias.
- Matplotlib: Nos servirá para graficar las iteraciones de la simulación

Estableceremos 5 constantes iniciales:

```
Nx, Ny = 400, 100 #Cantidad de celdas en X,Y
tau = 0.53 #Viscosidad cinetica
Nt = 7000 #Cantidad de iteraciones
Nl = 9 #Cantidad de nodos
plot_interval = 1000 #Intervalo de muestras al hacer la simulacion
```

Vectores de posición y pesos del modelo.

```
cxs = np.array([0, 0, 1, 1, 1, 0, -1, -1, -1])
cys = np.array([0, 1, 1, 0, -1, -1, -1, 0, 1])
pesos = np.array([4/9] + [1/9, 1/36]*4)
```

Aquí se definen las direcciones discretas y los pesos asociados del modelo D2Q9, la variable cxs son las direcciones cardinales de **X, Y = (0,1),(1,0),(0,-1),(-1,0) y** cxy son las direcciones en diagonal.

Los pesos son los coeficientes **wi** usados en el cálculo de equilibrio F_i^{eq}

Función de distancias

```
def distancia(x1, y1, x2, y2):
    return np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

Este realiza el calculo de la distancia euclidiana entre dos puntos, básicamente verifica si un punto está dentro del obstáculo cilíndrico.

Función de crear obstáculo

```
def crear_obstaculo_cilindrico(Nx, Ny, radio=13):
    cilindro = np.full( shape: (Ny, Nx),  fill_value: False)
    cx, cy = Nx // 4, Ny // 2
    for y in range(Ny):
        for x in range(Nx):
            if distancia(cx, cy, x, y) < radio:
                cilindro[y, x] = True
    return cilindro
```

Crea una mascara boleana que representa un cilindro solido dentro del espacio fasico, este se usa para reflejar partículas y anular el flujo en esta zona, esto se hace porque como mencionamos antes el espacio fasico no representa un espacio físico convencional, sino que representa un plano donde los eventos físicos no tienen límites. Por lo que si queremos representar el paso del fluido en un cilindro no colocamos el cilindro, sino que se representa como un limite donde el flujo no debe de pasar.

Función de distribución F

```
def inicializar_F(Ny, Nx, Nl):
    F = np.ones((Ny, Nx, Nl)) + 0.01 * np.random.randn(Ny, Nx, Nl)
    F[:, :, 3] = 2.3  # Flujo de entrada constante
    return F
```

Inicializa la distribución $\mathbf{F}$ para todas las direcciones con pequeñas perturbaciones y un flujo preferente hacia la derecha.

Aplicar las fronteras

```
def aplicar_fronteras(F):
    F[:, -1, [6, 7, 8]] = F[:, -2, [6, 7, 8]]  # Salida
    F[:, 0, [2, 3, 4]] = F[:, 1, [2, 3, 4]]    # Entrada
```

Esta es una función simple que se encarga de marcar donde entra el flujo (Izquierda) y la salida del mismo (Derecha)

función de colisión

```
def colision(F, rho, ux, uy):
    Feq = np.zeros_like(F)
    for i, cx, cy, w in zip(range(Nl), cxs, cys, pesos):
        cu = cx * ux + cy * uy
        Feq[:, :, i] = rho * w * (
            1 + 3 * cu + 4.5 * cu**2 - 1.5 * (ux**2 + uy**2)
        )
    return F - (1 / tau) * (F - Feq)
```

Anteriormente asignamos los límites de la simulación, pero ahora lo que vamos a hacer es marcar las partes donde se hará la colisión con el fluido y de esta manera poder generar el arrastre que tiene el fluido sobre el cilindro.

Streaming de partículas

```python
def streaming(F):
    for i, cx, cy in zip(range(Nl), cxs, cys):
        F[:, :, i] = np.roll(F[:, :, i], cx, axis=1)
        F[:, :, i] = np.roll(F[:, :, i], cy, axis=0)
    return F
```

Esta parte es la responsable de generar la simulación de las partículas en cada dirección desplazando los valores en la matriz.

Función de colisión: La función más importante

```python
def colision(F, rho, ux, uy):
    Feq = np.zeros_like(F)
    for i, cx, cy, w in zip(range(Nl), cxs, cys, pesos):
        cu = cx * ux + cy * uy
        Feq[:, :, i] = rho * w * (
            1 + 3 * cu + 4.5 * cu**2 - 1.5 * (ux**2 + uy**2)
        )
    return F - (1 / tau) * (F - Feq)
```

En el modelo LBM, la función colisión calcula la nueva distribución de partículas **Fi** en cada celda del dominio después de que colisionan entre sí. Las partículas tienden a acercarse a un estado de equilibrio conocido como F_i^{eq}, que depende de la densidad y la velocidad local.

$$F_i^{eq} = pw_i \left(1 + 3(c_i * u) + \frac{9}{2}(c_i * u)^2 - \frac{3}{2}u^2\right)$$

Esta fórmula representa cómo debería comportarse el sistema si estuviera en equilibrio, en base a su densidad y velocidad.

Por lo que el codigo simplemente recorre cada dirección del modelo D2Q9 en sus componentes cys, cxs y sus pesos y para ser especifico en la siguiente parte:

```
Feq[:, :, i] = rho * w * (
    1 + 3 * cu + 4.5 * cu**2 - 1.5 * (ux**2 + uy**2)
```

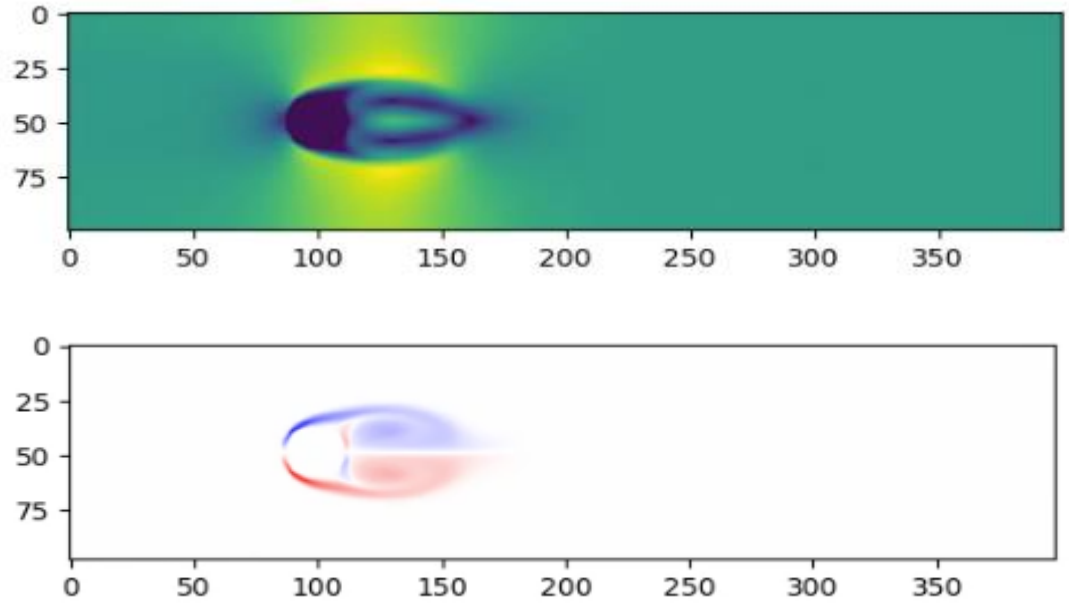
Es donde se implementa la función F_i^{eq} y de esta manera logramos mostrar como las partículas del fluido interactúan entre si y distribuyen su energía alrededor del cilindro.

Por lo que finalmente podemos realizar la simulación del fluido, en esta función se inicializa el dominio del espacio fasico, simula cada paso de tiempo y aplica las fases del codigo como ya hermosa visto, esto lo genera en base a las iteraciones por lo que encada iteración genera el siguiente frame con la informacion del frame que esta mostrando en el primer momento.

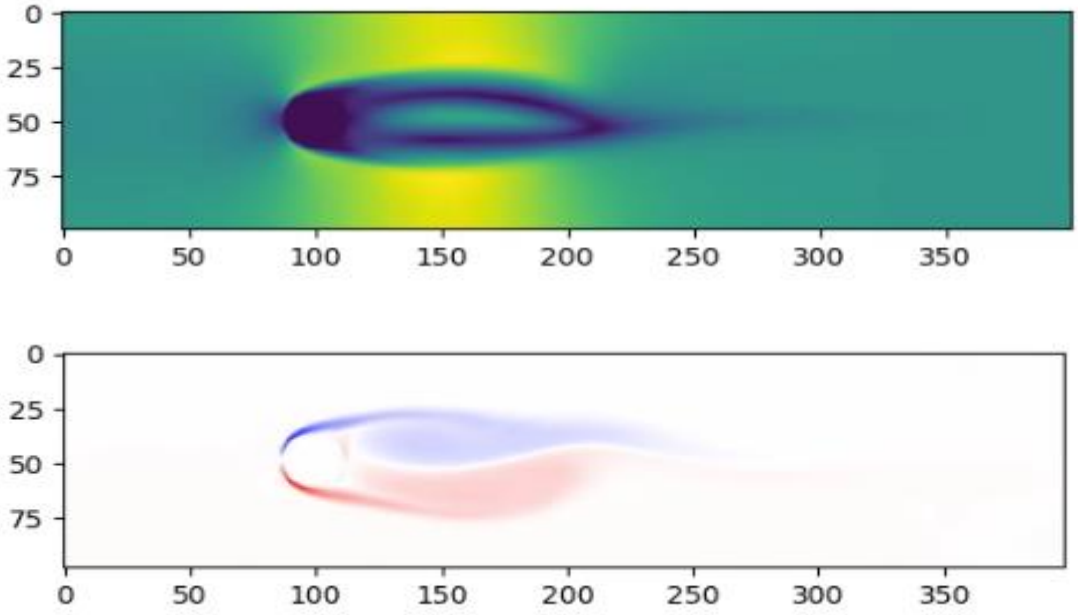
```
def simular():
    F = inicializar_F(Ny, Nx, Nl)
    cilindro = crear_obstaculo_cilindrico(Nx, Ny)

    for it in range(Nt):
        print(f"Iteración: {it}")

        aplicar_fronteras(F)
        F = streaming(F)

        # Reflejo de partículas en el obstáculo
        bndryF = F[cilindro, :]
        bndryF = bndryF[:, [0, 5, 6, 7, 8, 1, 2, 3, 4]]
        F[cilindro, :] = bndryF

        # Densidad y velocidades
        rho = np.sum(F, axis=2)
        ux = np.sum(F * cxs, axis=2) / rho
        uy = np.sum(F * cys, axis=2) / rho
        ux[cilindro], uy[cilindro] = 0, 0

        F = colision(F, rho, ux, uy)

        if it % plot_interval == 0:
            plt.imshow(np.sqrt(ux**2 + uy**2), cmap='plasma')
            plt.title(f"Velocidad en t={it}")
```

Con esto terminado podemos ver las simulaciones de los vórtices de Karman:

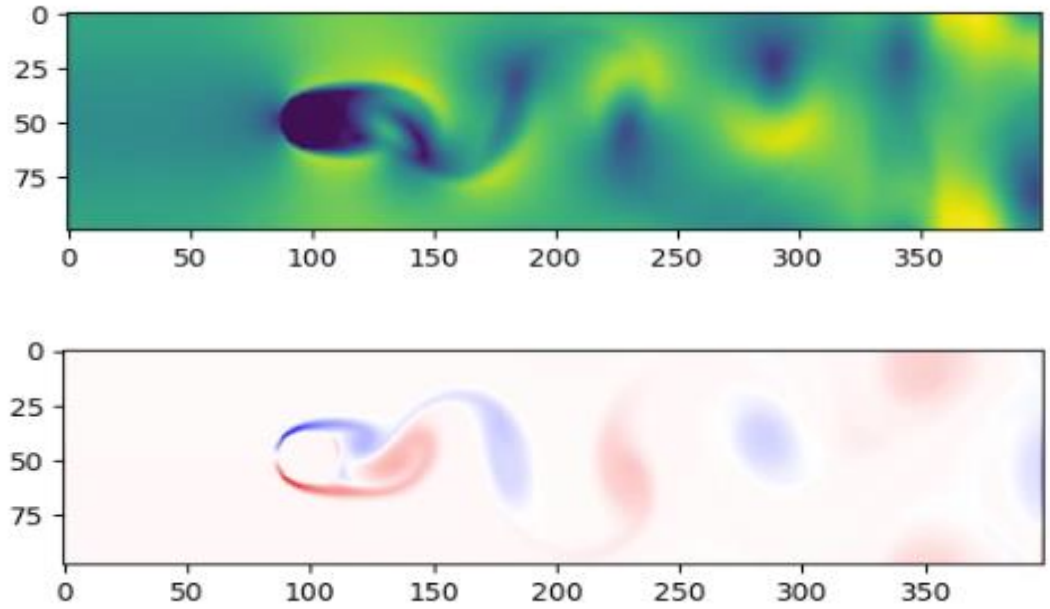
1000 iteraciones:



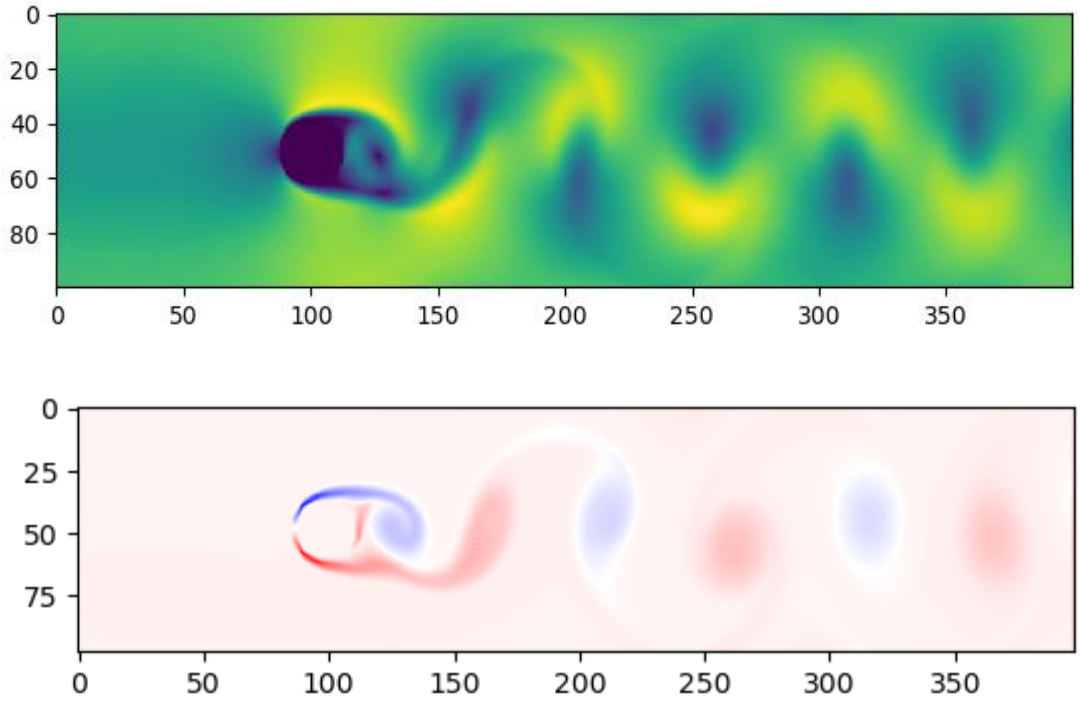
2500 iteraciones:



5000 iteraciones:



7000 iteraciones:

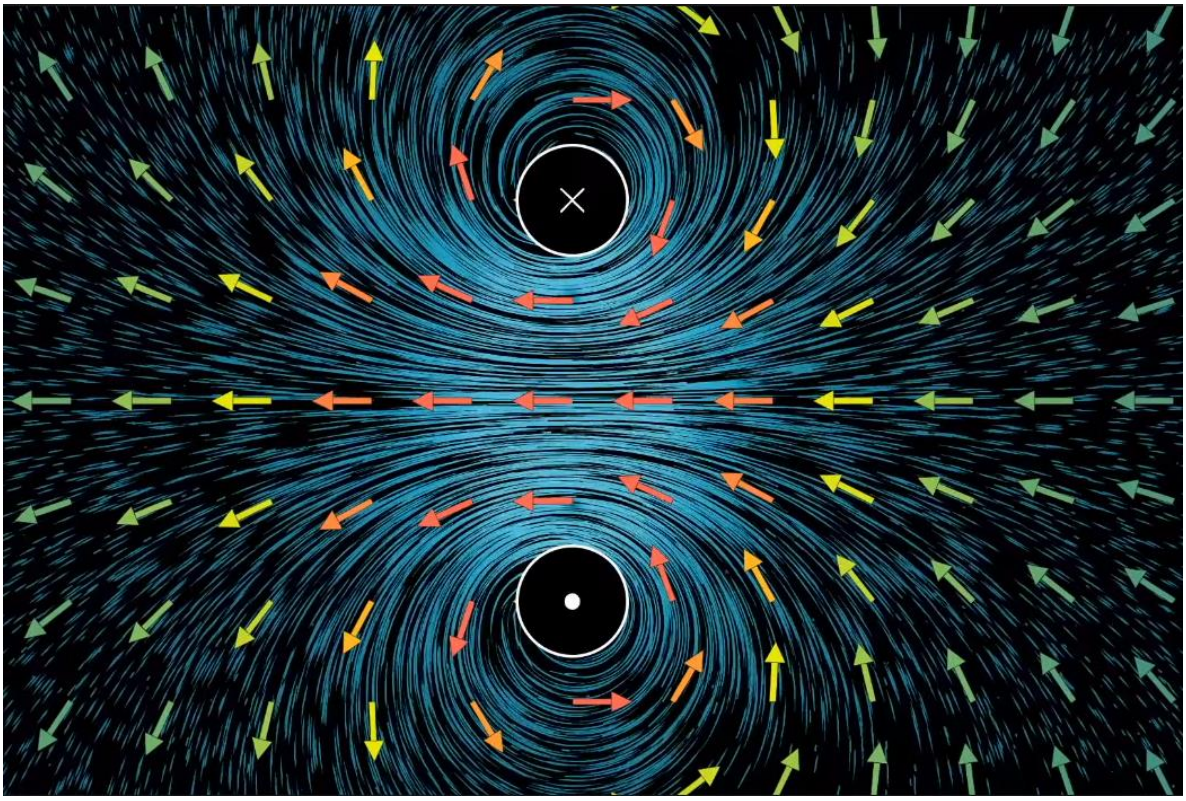


Conclusión

El poder de la matemática aplicada ha llegado a tal punto que podemos describir nuestra realidad de una manera muy concreta, con el avance de la velocidad de las computadoras a través de la reducción de tamaño de los transistores nos ha permitido modelar cada vez mas rápido, pero a pesar de esto los modelos matemáticos que tenemos disponibles son cada vez mas complicados de analizar y para que las computadoras puedan procesar esta gran cantidad de información, necesitamos sistemas de ecuaciones mas sencillos o al menos una forma de optimización clara.

En el caso de la dinámica de fluidos a pesar de que la demostración de Navier-Stokes se sigue investigando la realidad es que en muchos aspectos se le esta considerando como una ecuación obsoleta sobre todo en tres dimensiones, y que ya esta siendo reemplazada con otros métodos como el que se investigo en el presente ensayo, el método de Enrejado de Boltzmann.

A pesar de que las ecuaciones de Navier-Stokes están destinadas a reinar la dinámica de los fluidos, en realidad parece que mas bien abrieron un mundo de ecuaciones no descubiertas que si estarán destinadas a reinar la dinámica de fluidos.



Código fuente completo

```
import numpy as np
import matplotlib.pyplot as plt

# Tamaño del dominio
Nx, Ny = 400, 100

# Parámetros del modelo
tau = 0.53            # Tiempo de relajación (viscosidad del
fluido)
Nt = 7000             # Número total de iteraciones
Nl = 9                # Número de direcciones (modelo D2Q9)
plot_interval = 1000  # Intervalo de visualización

# Direcciones del modelo D2Q9 en X y Y
cxs = np.array([0, 0, 1, 1, 1, 0, -1, -1, -1])
cys = np.array([0, 1, 1, 0, -1, -1, -1, 0, 1])

# Pesos de cada dirección (w_i) en la función de equilibrio
pesos = np.array([4/9] + [1/9, 1/36]*4)

# ---------------------------------------------
# FUNCIONES AUXILIARES
# ---------------------------------------------

def distancia(x1, y1, x2, y2):
    """
    Calcula la distancia euclidiana entre dos puntos.
    Se usa para determinar si un punto está dentro del
cilindro.
    """
    return np.sqrt((x2 - x1)**2 + (y2 - y1)**2)

def crear_obstaculo_cilindrico(Nx, Ny, radio=13):
    """
    Crea una máscara booleana que representa un cilindro
sólido.
    El cilindro está centrado en (Nx/4, Ny/2) y tiene un radio
dado.
    """
    cilindro = np.full((Ny, Nx), False)
    cx, cy = Nx // 4, Ny // 2
    for y in range(Ny):
        for x in range(Nx):
```

```
            if distancia(cx, cy, x, y) < radio:
                cilindro[y, x] = True
    return cilindro

def inicializar_F(Ny, Nx, Nl):
    """
    Inicializa la distribución F con valores base y
perturbaciones aleatorias.
    Se impone un flujo inicial hacia la derecha (dirección 3).
    """
    F = np.ones((Ny, Nx, Nl)) + 0.01 * np.random.randn(Ny, Nx,
Nl)
    F[:, :, 3] = 2.3  # flujo inicial hacia la derecha
    return F

def aplicar_fronteras(F):
    """
    Aplica condiciones de frontera:
    - Copia valores de la segunda columna a la primera
(entrada).
    - Copia valores de la penúltima a la última (salida).
    """
    F[:, -1, [6, 7, 8]] = F[:, -2, [6, 7, 8]]
    F[:, 0, [2, 3, 4]] = F[:, 1, [2, 3, 4]]

def streaming(F):
    """
    Desplaza (streaming) las distribuciones F a lo largo de sus
respectivas direcciones.
    Se utiliza np.roll para simular el movimiento de
partículas.
    """
    for i, cx, cy in zip(range(Nl), cxs, cys):
        F[:, :, i] = np.roll(F[:, :, i], cx, axis=1)
        F[:, :, i] = np.roll(F[:, :, i], cy, axis=0)
    return F

def colision(F, rho, ux, uy):
    """
    Calcula la colisión mediante relajación hacia la función de
equilibrio F_eq,
    utilizando el modelo BGK. Esta función simula las
interacciones entre partículas.
    """
```

```
    Feq = np.zeros_like(F)
    for i, cx, cy, w in zip(range(Nl), cxs, cys, pesos):
        cu = cx * ux + cy * uy  # producto punto c_i · u
        Feq[:, :, i] = rho * w * (
            1 + 3 * cu + 4.5 * cu**2 - 1.5 * (ux**2 + uy**2)
        )
    return F - (1 / tau) * (F - Feq)

# ------------------------------------------
# FUNCION PRINCIPAL DE SIMULACIÓN
# ------------------------------------------

def simular():
    # Inicialización del campo de distribución y obstáculo
    F = inicializar_F(Ny, Nx, Nl)
    cilindro = crear_obstaculo_cilindrico(Nx, Ny)

    for it in range(Nt):
        print(f"Iteración: {it}")

        # Aplicar condiciones de frontera en entrada/salida
        aplicar_fronteras(F)

        # Mover las partículas (streaming)
        F = streaming(F)

        # Reflejo de partículas en el cilindro (bounce-back)
        bndryF = F[cilindro, :]
        bndryF = bndryF[:, [0, 5, 6, 7, 8, 1, 2, 3, 4]]  #
direcciones opuestas
        F[cilindro, :] = bndryF

        # Calcular variables macroscópicas: densidad y
velocidades
        rho = np.sum(F, axis=2)
        ux = np.sum(F * cxs, axis=2) / rho
        uy = np.sum(F * cys, axis=2) / rho

        # El fluido no puede moverse dentro del cilindro
        ux[cilindro] = 0
        uy[cilindro] = 0

        # Aplicar colisión para relajar hacia equilibrio
        F = colision(F, rho, ux, uy)
```

```
        # Visualización del campo de velocidad cada cierto
tiempo
        if it % plot_interval == 0:
            plt.imshow(np.sqrt(ux**2 + uy**2), cmap='plasma')
            plt.title(f"Velocidad en t={it}")
            plt.colorbar(label='Velocidad')
            plt.pause(0.01)
            plt.cla()

# Ejecutar simulación
if __name__ == "__main__":
    simular()
```

Fuentes:

- EcoFred. (s. f.). *Flujos laminares | Ecofred Sistemes*. Expertos en la fabricación e instalación de salas blancas y limpias. Recuperado 23 de mayo de 2022, de https://ecofred.com/es/soluciones/flujos-laminares

- Mora, P., Morra, G., & Yuen, D. A. (2019, 19 septiembre). *A concise python implementation of the lattice Boltzmann method on HPC for geo-fluid flow*. nsf.gov. Recuperado 22 de mayo de 2022, de https://par.nsf.gov/servlets/purl/10253868

- *Everything you need to know about the Lattice Boltzmann Method*. (2018, 26 agosto). FEA for All. Recuperado 22 de mayo de 2022, de https://feaforall.com/creating-cfd-solver-lattice-boltzmann-method/

- *How to Implement the Lattice Boltzmann Method (LBM) with a single relaxation time collision operator*. (2018, 26 agosto). FEA for All. Recuperado 22 de mayo de 2022, de https://feaforall.com/implementation-lattice-boltzmann-method-lbm/

- Fefferman, C. L. (s. f.). *EXISTENCE AND SMOOTHNESS OF THE NAVIER–STOKES EQUATION*. claymath.org. Recuperado 22 de mayo de 2022, de https://www.claymath.org/sites/default/files/navierstokes.pdf

- Stam, J. (2003, 6 marzo). *Real-Time Fluid Dynamics for Games*. graphics.cs.cmu.edu. Recuperado 20 de mayo de 2022, de http://graphics.cs.cmu.edu/nsp/course/15-464/Fall09/papers/StamFluidforGames.pdf

- Wagner, A. J. & Departamento de fisica de la universidad de Dakota del Norte. (2008, 1 marzo). *A Practical Introduction to the Lattice Boltzmann Method*. ndsu.edu. Recuperado 29 de mayo de 2022, de https://www.ndsu.edu/fileadmin/physics.ndsu.edu/Wagner/LBbook.pdf