Métodos Numéricos para la Ciencia e Ingeniería
Informe Tarea 7

Benjamín Guerrero
18.862.370-0
10 de Noviembre, 2015

1. Introducción

Esta tarea pide integrar las ecuaciones para ρ y para v, en un fluido compresible en una dirección, sin viscosidad ni gravedad, usando el método de las características.

Las ecuaciones son las siguientes:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0$$
$$\rho \frac{\partial v}{\partial t} + \rho v \frac{\partial v}{\partial x} = -\frac{\partial P}{\partial x}$$
$$P = A\rho^{\gamma}$$

Aquí, γ es 5/3, $A\left(x \leq \frac{1}{3}\right) = 4$, y $A\left(x > \frac{1}{3}\right) = 1$, con $x \in [0,1]$.

Se usarán las siguientes condiciones iniciales y de borde:

$$v(0,t) = 0$$
$$v(1,t) = 0$$
$$\rho(x > 0.1, 0) = 1$$
$$\rho(x \leq 0.1, 0) = 1 + 0.0RRR(1 + cos(10\pi x))$$

Aquí, $RRR = 3$ últimos números del RUT antes del dígito verificador (370).

Se debe integrar la ecuación hasta que el tiempo llegue a 1.2 en la mitad del intervalo entre cada x, y se debe dividir x en 10000 segmentos iguales.

2. Procedimiento

Primero, se ve la ecuación. Se aprecia que $\frac{\partial P}{\partial \rho} = A\gamma\rho^{\gamma-1} = \frac{\gamma P}{\rho}$. Se puede realizar el siguiente cambio de variable: $c^2 = \frac{\partial P}{\partial \rho} = \frac{\gamma P}{\rho}$.

Así, la segunda ecuación se puede reescribir de la siguiente forma (reemplazando y usando regla de la cadena):

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + \frac{c^2}{\rho}\frac{\partial \rho}{\partial x} = 0$$

Ahora, vamos a aplicar el método de las características. Para facilitar la redacción, se va a asumir que, para toda función F, $F_x = \frac{\partial F}{\partial x}$.

Así, podemos escribir $d\rho$ y dv de la siguiente manera:

$$d\rho = \rho_x dx + \rho_t dt$$

$$dv = v_x dx + v_t dt$$

Usando las ecuaciones, podemos reescribir estos términos así:

$$d\rho = \rho_x dx - (v\rho_x + \rho v_x)dt$$

$$dv = v_x dx - (vv_x + \frac{c^2}{\rho}\rho_x)dt$$

De la primera de estas ecuaciones se puede deducir que:

$$\rho_x = \frac{d\rho + \rho v_x dt}{dx - vdt}$$

Y reemplazando en la segunda:

$$\rho(dx - vdt)dv = \rho[(dx - vdt)^2 - c^2dt^2]v_x - c^2 d\rho dt$$

Así, la ecuación ya no depende de ρ_x. Para que no dependa de v_x, se asume que:

$$(dx - vdt)^2 - c^2dt^2 = 0$$

Luego, las características van a ser:

$$\frac{dx}{dt} = v \pm c$$

Y con esto:

$$c\frac{d\rho}{\rho} \pm dv = 0$$

Por lo tanto, las ecuaciones que hay que integrar son:

$$dx = (v + c)dt\,, \qquad cd\rho + \rho dv = 0$$

$$dx = (v - c)dt\,, \qquad cd\rho - \rho dv = 0$$

A continuación, se integran estas ecuaciones. Se vio en clases que esto resulta en:

$$x_R - x_P = (v_P + c_P)(t_R - t_P) \ (1)$$

$$x_R - x_Q = \left(v_Q - c_Q\right)(t_R - t_Q) \ (2)$$

$$c_P(\rho_R - \rho_P) + (v_R - v_P)\rho_P = 0 \ (3)$$

$$c_Q\left(\rho_R - \rho_Q\right) - \left(v_R - v_Q\right)\rho_Q = 0 \ (4)$$

$$x_R^{n+1} - x_P = \frac{1}{2}(v_P + v_R^n + c_P + c_R^n)(t_R^{n+1} - t_P) \ (5)$$

$$x_R^{n+1} - x_Q = \frac{1}{2}\left(v_Q + v_R^n - c_Q - c_R^n\right)(t_R^{n+1} - t_Q) \ (6)$$

$$(c_P + c_R^n)(\rho_R^{n+1} - \rho_P) = -(v_R^{n+1} - v_P)(\rho_R^n + \rho_P) \ (7)$$

$$\left(c_Q + c_R^n\right)\left(\rho_R^{n+1} - \rho_Q\right) = \left(v_R^{n+1} - v_Q\right)(\rho_R^n + \rho_Q) \ (8)$$

En el que R es el punto en donde se cruzan las rectas características que salen del punto P y del punto Q. Las primeras 4 ecuaciones son aproximaciones de primer orden, y las otras 4 son de segundo orden.

Ahora, en el caso actual, se define c^* como c^2:

$$c^*\left(x \leq \frac{1}{3}\right) = \frac{20\rho^{\frac{2}{3}}}{3}$$

$$c^*\left(x > \frac{1}{3}\right) = \frac{5\rho^{\frac{2}{3}}}{3}$$

Con esto, se empieza el programa. Se crea un programa llamado *fluido.py*, que realizará el programa. Primero, se crea una clase *Fluido* que tendrá todas las características que necesitamos. La clase se inicia definiendo las condiciones iniciales y de borde dadas, y definiendo c^*. También se inicia la condición *paso par* como True (se recuerda que los pasos pares son diferentes a los impares, se describirá después). Se divide x en 10000 segmentos iguales.

Luego, se define *paso_izquierdo*, en el que se toma el punto Q, y se calcula una primera aproximación de R usando las condiciones de borde (x=0) y las ecuaciones de aproximación de primer orden usando Q ((2) y (4)). Luego, se define c usando la primera ecuación ($x \leq \frac{1}{3}$) y se empieza a iterar para calcular R más precisamente, usando las ecuaciones de aproximación de segundo orden para Q ((6) y (8)). Esto se hace hasta las 1000 iteraciones o hasta que converja (o sea, que la aproximación sea muy exacta). Luego, los valores calculados para R se retornan.

Similarmente, se define *paso_derecho*, que toma un punto P. Las únicas diferencias son que se usan las condiciones de borde derechas (x=1), se usan las ecuaciones que usan P en lugar de Q (1, 3, 5 y 7), y c se define usando que ($x > \frac{1}{3}$). Esta función también retorna un R.

Para las 2 módulos anteriores, se añade una condición llamada *ver_convergencia* que chequea si se usa la iteración de convergencia o no. Si es True, se usa la iteración. Si es False, R se define como la primera aproximación.

Ahora, se define *paso_principal* (que será el más usado), que recibe dos puntos (P y Q), y calcula R a partir de esos puntos. Primero, se realiza una aproximación de primer orden combinando (1), (2), (3) y (4) (si a la ecuación (1) se le resta (2), se anula x_R y se puede sacar t_R). Luego, se saca c (se ve el valor de x para calcular c apropiadamente), y se ve si se quiere hacer la convergencia o no (usando *ver_convergencia*). Después, se itera y se usa la aproximación de segundo orden, combinando las ecuaciones 5 a la 8, hasta que converja.

Ahora, veamos el paso temporal.

Primero, se recuerda que si se pasa de un tiempo par a uno impar, se sigue las siguientes identificaciones:

$$P = (x_i^{2v}, t_i^{2v}), \qquad Q = (x_{i+1}^{2v}, t_{i+1}^{2v}), \qquad R = (x_i^{2v+1}, t_i^{2v+1})$$

En el otro caso (paso impar a par) es:

$$P = (x_{i-1}^{2v-1}, t_{i-1}^{2v-1}), \qquad Q = (x_i^{2v-1}, t_i^{2v-1}), \qquad R = (x_i^{2v}, t_i^{2v})$$

Ahora, se crea el módulo *avance_t*, que describe el avance en un paso temporal siguiendo lo descrito anteriormente. Se itera en el espacio x y se calculan todos los valores de R para ese paso temporal (usando los métodos definidos anteriormente), y se insertan a una lista llamada L. Luego, por conveniencia, se ordenan los valores x, t, v, ρ, c en ese orden, y se cambia el estado de *paso par* para seguir con el otro método. Cabe destacar que en los pasos impares se encuentran los bordes, se debe calcular R usando los métodos convenientes.

Finalmente, el método *estado_actual* simplemente retorna el estado actual del fluido.

Ahora, se crea el Main del programa. Primero se define F como la clase Fluido, y se extrae su estado actual. Luego, se crea una lista vacía llamada *Datos*, y se añade E a esta lista. Se inician el tiempo actual, y un contador.

Luego, se itera el tiempo hasta que llegue a 1.2, tal que contador sube de uno en uno en cada iteración. En cada iteración, se hace un paso temporal y se extrae su estado. Cada 100 iteraciones, el estado actual se guarda en *Datos*. Una vez hechas las iteraciones, se guarda *Datos* como un archivo .npy.

3. Resultados

Desafortunadamente, los resultados no fueron satisfactorios. Al correr el programa, se dejo correr este por más de 30 minutos, sin que este pudiera terminar. Se tuvo que terminar el programa manualmente.

4. Conclusión

Los errores presentes en esta experiencia pueden deberse a varios factores, entre los que se incluyen:

Que el código encontrara una situación excepcional que lo llevaría a un loop infinito.

Que se encuentre un bug en el código que no se ha podido identificar.

Que el computador usado para esta tarea no sea lo suficientemente poderoso para correr el programa.

Que el código este hecho de manera que hay cálculos de más (o sea, que no sea eficiente).

Viendo el código, se puede tantear que los valores de la densidad y de la velocidad van a variar considerando el efecto rebote de x=1/3.

Lo más difícil de la experiencia fue deducir que hacer para hacer la iteración, por lo que creo que fue eso lo que causo más problemas y demoras.