Segunda práctica voluntaria — Métodos Matemáticos y Técn. Comp.
Profesor Francisco R. Villatoro — 24 de Noviembre de 1999

Métodos en Diferencias Finitas para Ecuaciones Parabólicas.

Las ecuaciones de Navier-Stokes (Navier (1827), Poisson (1831), Saint-Venant (1843) y Stokes (1849)) para la conservación de la masa y del momento para un líquido/gas compresible son

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0,$$

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = \rho \, \mathbf{f_e} - \nabla p + \mu \left(\Delta \mathbf{v} + \frac{1}{3} \nabla \cdot (\nabla \cdot \mathbf{v}) \right),$$

donde ρ, p y $\mathbf{v}$ son la densidad, presión y velocidad del fluido (5 incógnitas para 4 ecuaciones), $\mathbf{f_e}$ son las fuerzas externas aplicadas (por ejemplo, la gravedad) y μ es el coeficiente de viscosidad (que para los gases es muy pequeño). A estas ecuaciones hay que añadirle una ecuación para la conservación de la energía, que depende de las propiedades termodinámicas del fluido. Todas estas ecuaciones en derivadas parciales definen un sistema mixto hiperbólico-parabólico y son bastante difíciles de resolver numéricamente, aunque ya hay mucho estudiado al respecto.

Como modelo del comportamiento de las soluciones de la ecuación de Navier-Stokes en el régimen parabólico (fluido viscoso), sobre todo para verificar métodos numéricos, se utiliza la ecuación de Burgers unidimensional

$$\frac{\partial v}{\partial t} + v \, \frac{\partial v}{\partial x} = \mu \, \frac{\partial^2 v}{\partial x^2}.$$

Esta ecuación es mixta (hiperbólica para μ pequeño, parabólica para μ general) y C-integrable (linealizable mediante un cambio de variables) mediante la transformación de Hopf-Cole (Hopf (1950) y Cole (1951)),

$$v = -2\mu \frac{\phi_x}{\phi}, \qquad \phi_x = \frac{\partial \phi}{\partial x}.$$

Aplicando la transformación en dos pasos, primero hacemos

$$v = \psi_x, \qquad \psi_t + \frac{1}{2}\psi_x^2 = \mu \psi_{xx},$$

y luego introducimos

$$\psi = -2\mu \log \phi, \qquad \phi_t = \mu\, \phi_{xx},$$

con lo que obtenemos la ecuación (parabólica) del calor.

Como la solución del problema de valores iniciales de la ecuación del calor es fácil de obtener, también lo será la solución de la ecuación de Burgers. El problema de valores iniciales para la ecuación del calor con $\phi(x,0) = \Phi(x)$ tiene como solución,

$$\phi = \frac{1}{\sqrt{4\pi\mu t}} \int_{-\infty}^{\infty} \Phi(y)\, \exp\left(-\frac{(x-y)^2}{2\mu t}\right)\, dy,$$

y, por tanto, para el problema de valores iniciales de la ecuación de Burgers con $v(x,0) = F(x)$, tenemos

$$\phi(x,0) = \Phi(x) = \exp\left(-\frac{1}{2\mu}\int_0^x F(\eta)\, d\eta\right),$$

y como solución general

$$v(x,t) = \frac{\int_{-\infty}^{\infty} \frac{x-y}{t}\, e^{-G/2\mu}\, dy}{\int_{-\infty}^{\infty} e^{-G/2\mu}\, dy},$$

donde

$$G(y;x,t) = \int_0^y F(z)\, dz + \frac{(x-y)^2}{2t}.$$

Como ejemplo de solución exacta de la ecuación de Burgers, que utilizaremos para verificar los códigos numéricos a desarrollar, presentaremos la onda de tipo N, que se obtiene para

$$\phi = 1 + \sqrt{\frac{a}{t}}\, e^{-x^2/4\mu t},$$

que nos da como solución exacta

$$v_e(x,t) = -\frac{2\mu\phi_x}{\phi} = \frac{x}{t}\, \frac{\sqrt{a/t}\, e^{-x^2/4\mu t}}{1 + \sqrt{a/t}\, e^{-x^2/4\mu t}},$$

1. Calcula la condición inicial $v_e(x,0)$. ¿Cómo evitas la división por cero? Haz un dibujo de dicha solución. Escribe un programa en Matlab que te permita calcular $v_e(x,t)$. Representa en Matlab la función $v_e(x,1)$ en función de μ para $a=1$. Comenta el comportamiento de esta solución de la ecuación de Burgers. ¿Cómo afecta el parámetro a? ¿Cómo afecta μ? ¿Puedes explicar físicamente el significado de estos dos parámetros?

2. Vamos a desarrollar métodos numéricos para la resolución del problema de valores iniciales y de contorno (no lineal)

$$\frac{\partial v}{\partial t} + v\,\frac{\partial v}{\partial x} = \mu\,\frac{\partial^2 v}{\partial x^2}, \qquad t \in [1,10], \quad x \in [-40,40],$$

$$v(x,1) = v_e(x,1), \qquad v(-20,t) = v(20,t) = 0,$$

con condiciones de contorno de Dirichlet homogéneas. La solución exacta de este problema se puede calcular mediante una serie, pero nos conformaremos con una aproximación a ella, la solución del problema de valores iniciales $v_e(x,t)$, ya que $v_e(-40,t) \approx 10^{-18}$ es del orden de los errores de redondeo. Como mallado utilizaremos

$$x_j = -40 + j\,h, \qquad j = 0,1,\ldots,N, \quad h \equiv \Delta x,$$

$$t_n = 1 + n\,k, \qquad n = 0,1,\ldots,M, \quad k \equiv \Delta t.$$

3. *Método de Euler Explícito.* Escribe un programa de Matlab que implemente el método $O(k,h^2)$

$$\frac{V_j^{n+1} - V_j^n}{k} + V_j^n\,\frac{V_{j+1}^n - V_{j-1}^n}{2h} = \mu\,\frac{\delta^2}{h^2}V_j^n \equiv \mu\,\frac{V_{j+1}^n - 2V_j^n + V_{j-1}^n}{h^2},$$

$$V_j^0 = v_e(x_j,1), \qquad V_0^n = V_N^n = 0.$$

Verifica la consistencia de este método. Lineariza esta ecuación tomando $vv_x \approx v_0 v_x$. Determina mediante el método de Fourier o de von Neumann la condición de estabilidad lineal de este método en función de v_0. Elige una malla (valores h y k) que garanticen la estabilidad del método y compara el resultado que obtienes con la solución "exacta" $v_e(x,t)$. ¿Cómo son los errores? Fija k y estudia cómo varía la norma infinito del error, $\|v_e(x_j,t_n) - V_j^n\|_\infty$ en función de h (estudia varios valores de h que garanticen la estabilidad). Fija h y estudia el error variando k. ¿Cómo chequearías numéricamente el orden del método $O(k,h^2)$?

4. *Método Explícito de Cuarto Orden.* Escribe un programa de Matlab que implemente el método $O(k, h^4)$

$$\frac{V_j^{n+1} - V_j^n}{k} + V_j^n \frac{V_{j-2}^n - 8V_{j-1}^n + 8V_{j+1}^n - V_{j+2}^n}{12h} = \mu \frac{1}{h^2} \left(\delta^2 - \frac{\delta^4}{12} \right) V_j^n,$$

$$V_j^0 = v_e(x_j, 1), \qquad V_0^n = V_1^n = V_{N-1}^n = V_N^n = 0,$$

y contesta a las mismas cuestiones que en el apartado anterior.

5. *Método de Euler Implícito.* Escribe un programa de Matlab que implemente el método $O(k, h^2)$

$$\frac{V_j^{n+1} - V_j^n}{k} + V_j^{n+1} \frac{V_{j+1}^{n+1} - V_{j-1}^{n+1}}{2h} = \mu \frac{\delta^2}{h^2} V_j^{n+1},$$

$$V_j^0 = v_e(x_j, 1), \qquad V_0^n = V_N^n = 0.$$

Utiliza el método de Newton para sistemas de ecuaciones para resolver la ecuación no lineal que obtienes en cada paso de tiempo. Contesta a las mismas cuestiones que en el apartado anterior.

6. *Método de Crank-Nicolson.* Escribe un programa de Matlab que implemente el método $O(k^2, h^2)$

$$\frac{V_j^{n+1} - V_j^n}{k} + V_j^{n+1/2} \frac{V_{j+1}^{n+1/2} - V_{j-1}^{n+1/2}}{2h} = \mu \frac{\delta^2}{h^2} V_j^{n+1/2},$$

$$V_j^{n+1/2} = \frac{V_j^{n+1} + V_j^n}{2}, \qquad V_j^0 = v_e(x_j, 1), \qquad V_0^n = V_N^n = 0.$$

Utiliza el método de Newton para sistemas de ecuaciones para resolver la ecuación no lineal que obtienes en cada paso de tiempo. Contesta a las mismas cuestiones que en el apartado anterior.

7. *Método con Operadores Compactos.* Verifica mediante Taylor que la derivada de una función se puede calcular con cuarto orden de precisión $O(h^4)$ utilizando la fórmula de Padé,

$$W_j \equiv (v_x)_j \approx \frac{1}{2h} \frac{V_{j+1} - V_{j-1}}{1 + \delta^2/6}, \qquad W_{j+1} + 4W_j + W_{j-1} = \frac{3}{h} (V_{j+1} - V_{j-1}).$$

Escribe un programa de Matlab que implemente el método $O(k^2, h^4)$

$$\frac{V_j^{n+1} - V_j^n}{k} + V_j^{n+1/2} W_j^{n+1/2} = \mu \frac{1}{2h} \frac{V_{j+1}^{n+1/2} - V_{j-1}^{n+1/2}}{1 + \delta^2/6},$$

$$W_j^{n+1/2} = \frac{1}{2h} \frac{V_{j+1}^{n+1/2} - V_{j-1}^{n+1/2}}{1 + \delta^2/6},$$

$$V_j^0 = v_e(x_j, 1), \qquad V_0^n = V_N^n = 0.$$

Repite las cuestiones del apartado anterior. (ESTE APARTADO ES MÁS DIFÍCIL QUE LOS ANTERIORES).

8. Finalmente, compara los métodos anteriores en cuanto a dificultad de implementación, estabilidad, precisión, número de operaciones flotantes necesarias para obtener un error dado, etc.

NOTA IMPORTANTE: La solución exacta v_e te sirve para verificar que tu implementación de los métodos numéricos anteriores es correcta, si tu método no coincide (para h y k suficientemente pequeños y cumpliendo la condición de estabilidad) con la solución exacta es que has cometido un error de implementación. Verifica tu código con cuidado en dicho caso.