Tercera práctica voluntaria — Técnicas Numéricas

Francisco R. Villatoro y Carmen M. García — 6 de Noviembre de 2000

ENUNCIADO DE LA TERCERA PRACTICA (PUNTUACIÓN: 0.2)

Algoritmos iterativos de resolución de sistemas lineales en Matlab.

Considere la ecuación de onda de primer orden con velocidad constante $c = 1$ y condiciones de contorno periódicas

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial u(x,t)}{\partial x} = 0, \qquad (x,t) \in (0,2\pi) \times \mathbb{R}^+,$$

$$u(x,t) = u(x+2\pi,t), \qquad u(x,0) = f(x), \qquad x \in (0,2\pi), \quad t > 0.$$

Vamos a obtener una solución numérica de esta ecuación. Sea M el número de puntos de la malla espacial, dado de antemano, definiremos una malla en espacio y tiempo, $\mathcal{T} \equiv \{(x_m, t_n)\}$ como

$$x_m = m\,\Delta x, \qquad m = 0, 1, 2, \ldots M-1, \qquad \Delta x = \frac{2\pi}{M},$$

$$t_n = n\,\Delta t, \qquad n = 0, 1, 2, \ldots;$$

aproximaremos $u(x,t)$ en dicha malla por $u^n_m \approx u(x_n, t_m)$, y aplicaremos un método pseudo-espectral de Fourier con Crank-Nicolson en tiempo (que se estudiará en la asignatura de cuarto curso), con lo que se obtendrá el sistema lineal

$$\left(I + \frac{1}{2}\mathbb{F}^{-1}\,(\mathrm{i}\,K)\,\mathbb{F} \right)\,\tilde{u}^{n+1} = \left(I - \frac{1}{2}\mathbb{F}^{-1}\,(\mathrm{i}\,K)\,\mathbb{F} \right)\,\tilde{u}^{n},$$

donde $\mathrm{i} = \sqrt{-1}$, $I \in \mathbb{R}^{M\times M}$ es la matriz identidad ($(I)_{i,j} = \delta_{ij}$), K denota la matriz diagonal de componentes $(K)_{ii} = i - M/2$, $\tilde{u}^n$ es el vector con componentes $(\tilde{u}^n)_i = u^n_{i-1}$ y, $\mathbb{F}$ y su inversa $\mathbb{F}^{-1}$ son matrices con componentes

$$(\mathbb{F})_{i,j} = e^{-\mathrm{i}\,(i-M/2)\,(j-1)\,\Delta x}, \qquad (\mathbb{F}^{-1})_{i,j} = e^{\mathrm{i}\,(i-M/2)\,(j-1)\,\Delta x},$$

donde los índices de las matrices son $1 \le i, j \le M$.

Iterando a partir de $(\tilde{u}^0)_m = f(x_m)$, podemos determinar $\tilde{u}^{n+1}$ a partir de $\tilde{u}^n$ resolviendo el sistema lineal $A\,z = b$, donde (en componentes)

$$(A)_{ij} = \delta_{ij} + \frac{\mathrm{i}}{2}\sum_{k=1}^{M} e^{\mathrm{i}\,(i-M/2)\,(k-1)\,\Delta x}\,(k - \frac{M}{2})\,e^{-\mathrm{i}\,(k-M/2)\,(j-1)\,\Delta x},$$

$$(z)_i \equiv u^{n+1}_{i-1}, \qquad (b)_i = u^n_{i-1} - \frac{1}{2}\sum_{j=1}^{M}\sum_{k=1}^{M} e^{\mathrm{i}\,(i-M/2)\,(k-1)\,\Delta x}\,(k - \frac{M}{2})\,e^{-\mathrm{i}\,(k-M/2)\,(j-1)\,\Delta x}\,u^n_{j-1}.$$

1. Escribe un programa en Matlab que calcule las matrices $\mathbb{F}$ y $\mathbb{F}^{-1}$ definidas más arriba. Comprueba que dichas matrices son inversa la una de la otra. Dibuja una gráfica con el número de condición de la matriz $\mathbb{F}$ en función de M. ¿Está bien condicionada dicha matriz?

2. Escribe un programa en Matlab para resolver la ecuación de onda unidimensional $u_t + u_x = 0$ con condición inicial $f(x) = \sin^2(x)\cos(x)$. Utiliza el método de Matlab $z = A\backslash b$. ¿Cuál es la solución exacta de dicha ecuación? Dibuja la solución numérica en $t = 2$ para $M = 128$ y los $\Delta t = 0.1, 0.05$, y 0.01. Dibuja el error entre la solución exacta y la numérica para dichos pasos de tiempo. ¿Qué norma has utilizado para definir el error? ¿Por qué?

3. Escribe una función que aplique de forma matricial el método iterativo de Gauss-Jacobi para resolver un sistema lineal. ¿Se puede aplicar dicho método a la matriz del sistema lineal del ejercicio anterior? ¿Por qué? Dibuja el número de condición de la matriz de iteración del método en función de M (para al menos 25 valores). Si el método converge, úsalo en el programa que has implementado en el apartado anterior. ¿Qué condiciones utilizas para chequear la convergencia del método de Gauss-Jacobi? ¿Por qué? Compara los resultados obtenidos con los del apartado anterior. Si iteras Gauss-Jacobi solamente una vez, antes de que converja, ¿cómo son los resultados que obtienes para $t = 2$ para los tres pasos de tiempo Δt anteriores? Justifica tus respuestas.

4. Repite el apartado anterior con el método de Gauss-Seidel.

5. Repite el apartado anterior con el método de sobrerrelajación sucesiva (SOR) basado en el método de Gauss-Seidel. Además, dibuja el radio espectral de la matriz de convergencia del método en función de w. ¿Cuándo converge?. Determine el w óptimo a partir de dicho dibujo. ¿Cómo determinas numéricamente la tasa de convergencia del método (utiliza sólo los iterados)? ¿Cómo determinas numéricamente el w óptimo utilizando sólo la tasa de convergencia numérica?

6. ¿Es aplicable el método del gradiente conjugado a la matriz de nuestro problema? En su caso aplícalo utilizando la función de Matlab `cgs`. De los algoritmos generalizados del gradiente conjugado que incluye Matlab `bicg`, `bicgstab`, `gmres`, `pcg` y `qmr`, ¿cuáles convergen para la matriz de nuestro problema? ¿Cuál es el más rápido?

APÉNDICE A LA PRÁCTICA : CORRECCIÓN DE ENUNCIADO

Justificación del algoritmo espectral utilizado en la práctica.

Para resolver la ecuación de onda lineal de primer orden vamos a desarrollar un método espectral. Los detalles son los siguientes:

Sea $\tilde{u}^n$ un vector con componentes $(\tilde{u}^n)_i = u^n_{i-1}$. Definiremos su transformada discreta de Fourier, siguiendo la notación de Matlab, como

$$\tilde{U}^n = \mathbb{F}\{\tilde{u}^n\}, \qquad (\tilde{U}^n)_j = \sum_{i=1}^{M} \exp(-\mathrm{i}\, 2\pi\,(j-1)\,(i-1))\,(\tilde{u}^n)_i, \qquad j = 1, 2, \ldots M,$$

que se puede escribir matricialmente como

$$\tilde{U}^n = \mathbb{F}\,\tilde{u}^n, \qquad (\mathbb{F})_{i,j} = \exp(-\mathrm{i}\, 2\pi\,(j-1)\,(i-1)),$$

y su transformada discreta de Fourier inversa como

$$\tilde{u}^n = \mathbb{F}^{-1}\{\tilde{U}^n\}, \qquad (\tilde{u}^n)_i = \frac{1}{M}\sum_{j=1}^{M} \exp(\mathrm{i}\, 2\pi\,(j-1)\,(i-1))\,(\tilde{U}^n)_j, \qquad i = 1, 2, \ldots M,$$

o, matricialmente, como

$$\tilde{u}^n = \frac{1}{M}\,\mathbb{F}^*\,\tilde{U}^n, \qquad (\mathbb{F}^*)_{i,j} = \exp(\mathrm{i}\, 2\pi\,(j-1)\,(i-1)).$$

Para calcular estas matrices en Matlab, escribiremos

```
x=0:0.1:2*pi;
u = sin(x).^2.*cos(x);
M=length(x); I=sqrt(-1); w=exp(-I*2*pi/M);

for i=1:M, for j=1:M, F(i,j) = w^((j-1)*(i-1)); end, end
Finv = F'/M;
```

Para calcular la derivada de una función, podemos calcular la transformada de Fourier, multiplicarla por la frecuencia $(\mathrm{i}\,K)$ y luego calcular la inversa de Fourier. De esta forma,

$$\frac{\partial \tilde{u}^n}{\partial x} = \mathbb{F}^{-1}\{\mathrm{i}\,K\,\mathbb{F}\{\tilde{u}^n\}\},$$

donde K se calcula en Matlab como la matriz diagonal

```
K = diag([ 0:N/2-1 , 0 , -(N/2-1:-1:1) ]);
```

Aplicando este método de cálculo de derivadas a nuestro problema, obtenemos el sistema de ecuaciones diferenciales

$$\frac{d\tilde{u}(t)}{dt} + \mathbb{F}^{-1}\{\mathrm{i}\,K\,\mathbb{F}\{\tilde{u}(t)\}\} = 0,$$

y aplicando un Método de Crank-Nicolson en tiempo

$$\frac{\tilde{u}^{n+1} - \tilde{u}^n}{\Delta t} + \frac{1}{M}\,\mathbb{F}^*\,\mathrm{i}\,K\,\mathbb{F}\,\frac{\tilde{u}^n + \tilde{u}^{n+1}}{2} = 0.$$

De esta forma obtenemos el sistema lineal de ecuaciones $A\,z = B\,\tilde{u}^n$, donde, $z \equiv \tilde{u}^{n+1}$ y, escrito en notación matricial de Matlab,

```
A = eye(M) +  dt/(2*M)*F'*(I*K)*F,
B = eye(M) -  dt/(2*M)*F'*(I*K)*F,
```

donde `dt` $\equiv \Delta t$.

Los apartados de la práctica se contestarán con estas nuevas A y F.

```
%%% Tercera práctica voluntaria de Técnicas Numéricas
%%%              APÉNDICE SEGUNDO A LA PRÁCTICA
%%%                  Francisco R. Villatoro
%
% para facilitar la realización, aquí tenéis un código Matlab

clear all

dt = 0.1; T=4;  %% tiempo máximo de integración numérica
N=ceil(T/dt); t = 0:dt:(N-1)*dt;

%%% IMPORTANTE: M debe ser un número par
M=128; x=(0:(2*pi)/(M-1):2*pi)';

u0 = sin(x).^2.*cos(x); u=u0; %%% condición inicial

I=sqrt(-1); w=exp(-I*2*pi/M);

for i=1:M, for j=1:M, F(i,j) = w^((j-1)*(i-1)); end, end Finv =
F'/M;

%%% Comprueba la FFT
U = F*u; Ufft = fft(u); u2 = Finv*U; uf = ifft(Ufft);

%%% Calculo de derivadas
K = diag([ 0:M/2-1 , 0 , -(M/2-1:-1:1) ]);

%%% Comprueba calculo de derivadas
uxf = ifft( I*K*fft(u)); ux = Finv*I*K*F*u;

%%% Método de Crank-Nicolson para u,t + u,x = 0
A = eye(M) + dt/2*Finv*I*K*F; B = eye(M) - dt/2*Finv*I*K*F;

%%% Imprimir sólo algunos valores de la solución
colors = 'ymcrgbk'; color = 1; clf plot
(x,real(u0),colors(color)); hold on; u = u0; for n=1:20,
 if (1/rcond(A)>10^7), 1/rcond(A), end
 u = A\(B*u);
 if (mod(n,4)==0),
  color = color+1; if (color>length(colors)), color = 1; end
  plot (x,real(u),colors(color));
 end
end

%%% Imprimir la solución completa
pause; clf; u = zeros(M,N); u(:,1)=u0; for n=2:N,
```

```
 u(:,n)=A\(B*u(:,n-1));
end mesh(t,x,real(u)); xlabel ('t'); ylabel('x');
axis([0,T,0,2*pi,-0.5,0.5]); pause; contour(t,x,real(u));
```