Ecuaciones en Derivadas Parciales y Análisis Numérico

Prácticas

Capítulo 3. Diferencias finitas para la ecuación del calor.

3.1 Resolviendo la ecuación del calor

Vamos a resolver la ecuación del calor utilizando un método de diferencias finitas. Como veremos, podemos aprovechar las ideas que utilizamos con las ecuaciones diferenciales ordinarias. Centrémonos en la ecuación del calor sin fuentes de calor:

$$u_t - k\,u_{xx} = 0$$

en una dimensión espacial, en el intervalo $[0,1]$, para tiempos entre 0 y t_f, con condiciones de contorno tipo Dirichlet:

$$u(0,t) = 0 \qquad u(1,t) = 0$$

y con condición inicial:

$$u(x,0) = x(1-x)$$

Para resolver la ecuación, la interpretamos como un sistema de ecuaciones diferenciales ordinarias (EDOs) en las que un valor inicial $u^0(x) = u(x,0)$ evoluciona según una EDO. Para ello es necesario utilizar una versión *discreta* de este valor inicial para poder representarlo en el ordenador mediante un vector. Descomponemos el intervalo $[0,\,1]$ en M partes iguales, obteniendo un mallado uniforme $x_1 = 0$, ..., $x_{m+1} = x_1 + m/M$, $x_{M+1} = 1$, y consideramos los valores de u sólo en esos puntos. Como dato inicial tomamos el vector:

$$\hat{u}^0 = (\hat{u}^0_m)_{m=1}^{M+1},\ \hat{u}^0_m = x_m\,(1-x_m)$$

Nuestro objetivo es encontrar valores $\hat{u}_m(t)$ que aproximen el valor de la función u en el punto x_m y en el instante t. Los valores de u en los puntos x_1 y x_{M+1} son datos del problema (los datos de contorno), de modo que sólo falta encontrar valores que aproximen u en los puntos desde x_2 hasta x_M (un total de $M-1$ puntos).

Representamos las derivadas espaciales mediante un operador discreto que actúe sobre vectores de $\mathbb{R}^{M-1}$. Definimos $h = 1/M$ y utilizamos diferencias centradas para aproximar la segunda derivada de u:

$$D(\hat{u})_m = \tfrac{1}{h^2}(\hat{u}_{m-1} - 2\hat{u}_m + \hat{u}_{m+1})$$

El operador D se representa por la matriz:

$$D(\hat{u}) = \tfrac{1}{h^2}\begin{pmatrix} -2 & 1 & 0 & & \\ 1 & -2 & 1 & & \\ & & \dots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \cdot \hat{u}$$

Observa con atención la primera y la última fila, donde se ha utilizado la condición de frontera

$$D(\hat{u})_2 = \tfrac{1}{h^2}(\hat{u}_1 - 2\hat{u}_2 + \hat{u}_3) = \tfrac{1}{h^2}(-2\hat{u}_2 + \hat{u}_3)$$
$$D(\hat{u})_M = \tfrac{1}{h^2}(\hat{u}_{M-1} - 2\hat{u}_M + \hat{u}_{M+1}) = \tfrac{1}{h^2}(\hat{u}_{M-1} - 2\hat{u}_M)$$

Ahora la EDO para $\hat{u}$ es:

$$\tfrac{d}{dt}\hat{u} = kD(\hat{u})$$
$$\hat{u}(0) = \hat{u}^0$$

y la podemos resolver de forma aproximada utilizando cualquiera de los métodos vistos para EDOs. Por ejemplo, si usamos el método de Euler explícito, obtenemos el algoritmo:

$$\hat{u}^{n+1} = \hat{u}^n + \mathrm{dt} \cdot kD(\hat{u}^n) = (I + \mathrm{dt} \cdot kD)(\hat{u}^n)$$

Escribimos un script de matlab que implementa la idea anterior. El script toma los datos relevantes del problema y devuelve vectores $\hat{u}^n$ que aproximan $\hat{u}(t_n)$ para algunos valores tiempos.

```
function [u_save,x,t_save]=calor(k, M, t_f, tsteps)
%function [u_save,x,t_save]=calor(k, M, t_f, tsteps)
%
%resuelve la ecuacion del calor u_t = k u_xx en [0,1] con dato inicial x(1-x)
%y condiciones de contorno tipo Dirichlet u(0,t)=u(1,t)=0
%
%Datos de entrada:
% k: coeficiente k de la ecuacion
% M: numero de partes iguales en que se descompone [0,1]
% t_f: instante de tiempo hasta el que calculamos la solucion
% tsteps: numero de partes iguales en que se descompone [0,t_f]

%Mallado
dt=t_f/tsteps;
tiempos=0:dt:t_f;
h=1/M;
x=0:h:1;

%Datos iniciales
xred=x(2:M); %quitamos los extremos del vector x
u=(xred.*(1-xred))'; %trasponemos para obtener un vector columna

%El operador D
%Usamos el comando diag(vector,k) para crear una matriz tridiagonal
Tridiag=diag(-2*ones(M-1,1))+diag(ones(M-2,1),1)+diag(ones(M-2,1),-1);
D=(1/h^2)*Tridiag;

%Datos de salida: en vez de guardar todos los datos intermedios, guardamos el
%vector u solo Nframes veces.
Nframes=5;
marca=floor(tsteps/(Nframes-1));
u_save=zeros(M+1,Nframes);
t_save=zeros(1,Nframes);
%Le ponemos las condiciones de frontera
u_save(1,:)=zeros(1,Nframes);
u_save(M+1,:)=zeros(1,Nframes);
%guardamos la posicion de partida
u_save(2:M,1)=u;
t_save(1)=0;

%Bucle principal
I=eye(M-1);
A=(I+dt*k*D);
for n=1:tsteps
    u=A*u;
    %Guardamos los valores de u para algunos tiempos
    if mod(n,marca)==0
```

```
      indice=1+n/marca;
      u_save(2:M,indice)=u;
      t_save(indice)=tiempos(n);
    end
end
```

3.2 Representar la solución

Al llamar al código anterior con datos concretos:

```
[u,x,t]=calor(.1, 20, 1, 1000);
```

recuperamos el vector con las posiciones `x`, los instantes de tiempo `t(1)`, ... ,`t(Nframes)` en que se ha guardado la solución, y los vectores `u(n)` que aproximan $\hat{u}(t(n))$ para esos tiempos. Para representar la solución en el instante `t(n)` escribimos:

```
plot(x,u(:,n));
```

Si queremos ver cómo evoluciona la temperatura, podemos dibujar todos los vectores `u(n)` a la vez:

```
plot(x,u)
```

Ejercicio: vuelve a ejecutar el código anterior con otra condición inicial distinta. Dibuja el resultado. ¿Qué ocurre si la condición inicial no satisface la condición de frontera?

3.3 Estabilidad del método

Vamos a calcular la solución hasta $t_f = 1$ con $M = 100$, tsteps $= 1000$, $k = 0.1$.
Para ello escribe:

```
[u,x,t]=calor(.1, 100, 1, 1000);
```

y dibuja la solución.
Como podemos comprobar, el resultado es altamente oscilatorio y la temperatura toma valores enormes, cosa que no es físicamente razonable. El problema está en la elección del paso temporal. Recordamos que para que el esquema numérico sea estable se debe cumplir

$$s = k\frac{dt}{h^2} < \frac{1}{2}$$

Si tomamos $M = 100$, el paso temporal debe verificar $dt < \frac{1}{2}\frac{1}{0.1}\left(\frac{1}{100}\right)^2 = 5 \cdot 10^{-4}$. Como ejercicio, puedes comprobar que en este caso la cota es exacta, y valores de dt ligeramente superiores producirán una solución inestable mientras que valores ligeramente inferiores producen una aproximación decente (el valor crítico de tsteps es 2000).

3.4 Un problema más complejo

Vamos a estudiar ahora una situación un poco más complicada.
La ecuación que vimos en el apartado 3.1 se puede usar para representar la temperatura en una barra de metal cuyo cuerpo está aislado y no recibe calor de ninguna fuente pero cuyos extremos se mantienen a temperatura cero.
El ejemplo que veremos ahora corresponde a una barra cuyos extremos están aislados y que recibe calor de una fuente externa. En concreto, queremos estudiar cómo se calienta una barra con temperatura inicial 0 y los extremos aislados cuando calentamos la mitad izquierda de la barra.

De nuevo estudiamos el problema en una dimensión espacial, en el intervalo [0, 1], para tiempos entre 0 y t_f. Para comenzar, debemos añadir una fuente de calor a la ecuación del calor:

$$u_t - k\,u_{xx} = f(t,x)$$

Tomaremos la fuente de calor constante en el tiempo y de intensidad 1 o 0 dependiendo de si estamos en la primera mitad de la barra o en la segunda:

$$f(t,x) = \begin{cases} 1 \text{ si } x < \frac{1}{2} \\ 0 \text{ si } x \geq \frac{1}{2} \end{cases}$$

Además de la fuente de calor, tenemos condiciones de contorno tipo Neumann homogéneas:

$$\frac{\partial u}{\partial x}(t,0) = \frac{\partial u}{\partial x}(t,1) = 0$$

que representan que no hay flujo de calor en los extremos de la barra.

Del mismo modo que antes, interpretamos la ecuación como un sistema de ecuaciones diferenciales ordinarias (EDOs) en las que el valor inicial $u^0(x) = u(x,0)$ evoluciona según una EDO. De nuevo buscamos representar una función $u(x)$ de una variable x mediante un vector. Para ello descomponemos el intervalo $[0,1]$ en M partes iguales, obteniendo un mallado uniforme $x_1 = 0$, ..., $x_{m+1} = x_1 + m/M$, $x_{M+1} = 1$, y consideramos los valores de u sólo en esos puntos. Tomamos como dato inicial el vector

$$\hat{u}^0_m = 0 \quad \forall m = 1, \ldots, M+1$$

Nuestro objetivo es encontrar valores $\hat{u}_m(t)$ que aproximen el valor de la función u en el punto x_m y en el instante t. Los valores en los puntos x_1 y x_{M+1} ya no son datos del problema, de modo que ésta vez tenemos que encontrar valores que aproximen u en los puntos desde x_1 hasta x_{M+1} (un total de $M+1$ puntos).

Representamos las derivadas espaciales mediante un operador discreto que actúe sobre vectores de $\mathbb{R}^{M+1}$. Definimos $h = 1/M$ y utilizamos la misma aproximación de antes para la segunda derivada de u

$$D(\hat{u}_m) = \tfrac{1}{h^2}(\hat{u}_{m-1} - 2\hat{u}_m + \hat{u}_{m+1})$$

Sin embargo, tendremos que trabajar un poco más para encontrar las ecuaciones en los extremos del intervalo. Utilizando una diferencia centrada podemos aproximar la condición en la frontera:

$$0 = \frac{\partial u}{\partial x}(x_1,t) \approx \tfrac{1}{2h}(u(x_1+h,t) - u(x_1-h,t))$$
$$0 = \frac{\partial u}{\partial x}(x_{M+1},t) \approx \tfrac{1}{2h}(u(x_{M+1}+h,t) - u(x_{M+1}-h,t))$$

$$0 = \tfrac{1}{2h}(u_2(t) - u^*_0(t))$$
$$0 = \tfrac{1}{2h}(u^*_{M+2}(t) - u_M(t))$$

Al hacerlo aparece la función u evaluada en el punto $x_1 - h$ (el punto x_0, en cierto sentido), que no pertenece al intervalo, pero este punto también aparece al utilizar la aproximación a $D^2(\hat{u})_1$, de modo que podemos cancelar ambas apariciones para obtener las aproximaciones:

$$D(\hat{u})_1 = \tfrac{1}{h^2}(\hat{u}^*_0 - 2\hat{u}_1 + \hat{u}_2) = \tfrac{1}{h^2}(-2\hat{u}_1 + 2\hat{u}_2)$$
$$D(\hat{u})_{M+1} = \tfrac{1}{h^2}(\hat{u}_M - 2\hat{u}_{M+1} + \hat{u}^*_{M+2}) = \tfrac{1}{h^2}(2\hat{u}_M - 2\hat{u}_{M+1})$$

De este modo, el operador D se representa por la matriz:

$$D(\hat{u}) = \frac{1}{h^2}\begin{pmatrix} -2 & 2 & 0 & & \\ 1 & -2 & 1 & & \\ & & \dots & & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{pmatrix} \cdot \hat{u}$$

La fuente de calor f es más fácil de representar: es sólo un vector de longitud $M+1$ con las primeras $\frac{M+1}{2}$ entradas iguales a 1 y las siguientes iguales a 0.

De este modo, la EDO para $\hat{u}$ es:

$$\begin{gathered} \frac{d}{dt}\hat{u} = kD(\hat{u}) + f \\ \hat{u}(0) = \hat{u}^0 \end{gathered}$$

Volvemos a usar el método de Euler explícito, que en este caso da lugar a:

$$\hat{u}^{n+1} = \hat{u}^n + dt \cdot (kD(\hat{u}^n) + f) = (I + dt \cdot kD)\hat{u}^n + dt \cdot f$$

Adaptamos el código anterior a la nueva situación:

```
function [u_save,x,t_save]=calor2(k, M, t_f, tsteps)
%function [u_save,x,t_save]=calor2(k, M, t_f, tsteps)
%
%resuelve la ecuacion del calor u_t = k u_xx + f en [0,1] con dato inicial 0
%y condiciones de contorno tipo Neumann homogeneas
%
%Datos de entrada:
% k: coeficiente k de la ecuacion
% M: numero de partes iguales en que se descompone [0,1]
% t_f: instante de tiempo hasta el que calculamos la solucion
% tsteps: numero de partes iguales en que se descompone [0,t_f]

%Mallado
dt=t_f/tsteps;
tiempos=0:dt:t_f;
h=1/M;
x=0:h:1;

%Datos iniciales
u=zeros(M+1,1);

%El operador D
%Usamos el comando diag(vector,k) para crear una matriz tridiagonal
Tridiag=diag(-2*ones(M+1,1))+diag(ones(M,1),1)+diag(ones(M,1),-1);
Tridiag(1,2)=2;
Tridiag(M+1,M)=2;
D=(1/h^2)*Tridiag;

%El vector f
mitad=floor((M+1)/2);
f=[ones(mitad,1); zeros(M+1-mitad,1)];

%Datos de salida: en vez de guardar todos los datos intermedios, guardamos el
%vector u solo Nframes veces.
Nframes=5;
marca=floor(tsteps/(Nframes-1));
u_save=zeros(M+1,Nframes);
```

```
t_save=zeros(1,Nframes);
%guardamos la posicion de partida
u_save(:,1)=u;
t_save(1)=0;

%Bucle principal
I=eye(M+1);
A=(I+dt*k*D);
b=dt*f;
for n=1:tsteps
    u=A*u+b;
    %Guardamos los valores de u para algunos tiempos
    if mod(n,marca)==0
      indice=1+n/marca;
      u_save(:,indice)=u;
      t_save(indice)=tiempos(n);
    end
end
```

Probemos el resultado:

```
[u,x,t]=calor2(.05, 20, 1, 1000);
plot(x,u)
```