Certamen #3 Mat-015 (Etapa 1)

2^{do} Semestre 2014

Victor Gonzalez Rodriguez
victor.gonzalezro@alumnos.usm.cl
2773029-9

10 de diciembre de 2014

$$(\mathbf{P}) \begin{cases} u_t - c^2 u_{xx} = f(x,t) & 0 < x < l,\ c > 0 \\ u(0,t) = u(l,t) = 0 & t > 0 \\ u(x,0) = g(x) & 0 < x < l \end{cases}$$

1. Planteo de esquema de diferencias finitas.

1.1. Planteo analítico.

Se nos dice que los valores a aproximar son denotados por $u(x_j, t_k) = u_j^k$, y que las derivadas se definen mediante las siguientes discretizaciones:

$$u_t(x_j, t_k) \approx \frac{u_j^k - u_j^{k-1}}{h_t}, \quad u_{xx} \approx \frac{u_{j+1}^k - 2u_j^k + u_{j-1}^k}{h_x^2}$$

Luego, podemos reescribir la EDP de manera discreta:

$$u_t - c^2 u_{xx} = f(x,t) \tag{1}$$

$$u_t = c^2 u_{xx} + f(x,t) \tag{2}$$

$$u_t(x_j, t_k) = c^2 u_{xx}(x_j, t_k) + f(x_j, t_k) \tag{3}$$

$$\frac{u_j^k - u_j^{k-1}}{h_t} = c^2 \frac{(u_{j+1}^k - 2u_j^k + u_{j-1}^k)}{h_x^2} + f_j^k \tag{4}$$

$$u_j^k - u_j^{k-1} = \frac{c^2 h_t}{h_x^2}(u_{j+1}^k - 2u_j^k + u_{j-1}^k) + h_t f_j^k \tag{5}$$

En **(5)** denotaremos la fracción de constantes como *lambda*: $\lambda = \dfrac{c^2 h_t}{h_x^2}$. Además, podemos notar que el problema tiene la forma de *atrás en el tiempo*, el cual se resuelve mediante métodos de matrices.

Luego, nuestro problema queda de la siguiente manera:

$$u_j^k - u_j^{k-1} = \lambda(u_{j+1}^k - 2u_j^k + u_{j-1}^k) + h_t f_j^k \tag{6}$$

$$u_j^{k-1} + h_t f_j^k = (-\lambda)u_{j-1}^k + (1+2\lambda)u_j^k + (-\lambda)u_{j+1}^k \tag{7}$$

Podemos mover el tiempo en una unidad: $(k-1) \to k$ y $k \to (k+1)$. Entonces:

$$u_j^k + h_t f_j^{k+1} = (-\lambda)u_{j-1}^{k+1} + (1+2\lambda)u_j^{k+1} + (-\lambda)u_{j+1}^{k+1} \tag{8}$$

Por lo tanto, en **(8)** tenemos discretizada la solución de la EDP original. Luego, las condiciones de borde se escriben como:

$$u(0,t) = 0 \;\Rightarrow\; u(x_o, t_j) = 0 \;\Rightarrow\; u_0^j = 0$$
$$u(l,t) = 0 \;\Rightarrow\; u(x_{n+1}, t_j) = 0 \;\Rightarrow\; u_{n+1}^j = 0$$

Y la condición inicial queda dada por: $u(x,0) = g(x) \Rightarrow u(x_k, t_0) = g(x_k) \Rightarrow u_0^k = g(x_k)$.

Adicionalmente, se dice que $\lambda < 1$, para que el problema se considere estable, y pueda converger.

En la ecuación **(8)**, se puede observar que las soluciones se pueden encontrar mediante un sistema de ecuaciones $Au = f_0$.

$$\begin{bmatrix} (1+2\lambda) & (-\lambda) & 0 & 0 & 0 & 0 \\ (-\lambda) & (1+2\lambda) & (-\lambda) & 0 & 0 & 0 \\ 0 & (-\lambda) & (1+2\lambda) & (-\lambda) & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & (-\lambda) & (1+2\lambda) & (-\lambda) \\ 0 & 0 & 0 & 0 & (-\lambda) & (1+2\lambda) \end{bmatrix} \begin{bmatrix} u_1^{k+1} \\ u_2^{k+1} \\ u_3^{k+1} \\ \vdots \\ u_{n_x-1}^{k+1} \\ u_{n_x}^{k+1} \end{bmatrix} = \begin{bmatrix} u_1^k + h_t f_1^{k+1} \\ u_2^k + h_t f_2^{k+1} \\ u_3^k + h_t f_3^{k+1} \\ \vdots \\ u_{n_x-1}^k + h_t f_{n_x-1}^{k+1} \\ u_{n_x}^k + h_t f_{n_x}^{k+1} \end{bmatrix}$$

1.2. Planteo del algoritmo.

- Generar un mallado que contenga las regiones para las cuales esta definido el problema.
- Aplicar valor inicial $u_0^k = g(x_k)$ y las condiciones de borde.
- Generar matriz A tridiagonal.
- Iterar sobre el tiempo, resolviendo cada sistema de ecuaciones.
- Entregar resultado.

2. Implementación en Matlab.

```
function calor(L,T,n,m,c)
    % u(x,t) = e^(-3t)sin(x) + xt
    % f(x,t) = x
    % g(x) = sin(x)

    % mallado
    dx=L/n;          dt=1/m;
    x = 0:dx:L;      t = 0:dt:T;

    lambda = c^2*dt/(dx^2);     % solo definido asi por comodidad

    f0 = zeros(n+1,1);          % array con valores iniciales
    U = zeros(n+1,m+1);         % matriz de resultado
    A = zeros(n+1,n+1);         % matriz tridiagonal de rigidez

    % condicion de borde y valor inicial
    for i=1:n
        f0(i,1) = sin(x(1,i));  % u(x,0) = g(x)
    end
    f0(1,1) = 0;                % u(0,t) = 0
    f0(n+1,1) = 0;              % u(l,t) = 0

    % llenar la matriz tridiagonal con los coeficientes
    for i= 1:1:n
        A(i,i) = 1 + 2*lambda;
        A(i,i+1) = -lambda      ;
        A(i+1, i) = -lambda     ;
    end
```

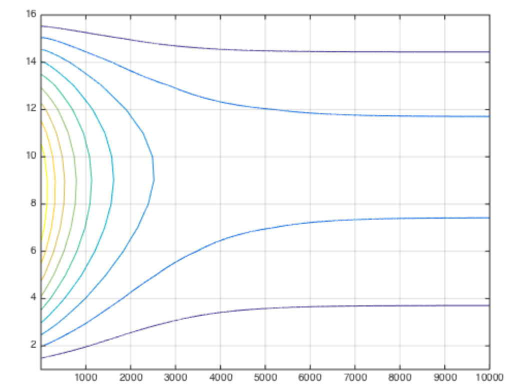
```
A(n+1,n+1) = 1+2*lambda;    % condicionar el ultimo valor

U(:,1) = f0 + x'*dt;        % inicializacion de algoritmo
for j = 1:1:m
    U(:,j+1) = A\(f0+ x'*dt); % Resolvemos Au=f0 para u
    f0 = U(:,j+1);
end

% graficamos para comparar
figure
contour3(U)

[xx,tt]=meshgrid(x,t);
exact=exp(-3.*tt).*sin(xx) + xx.*tt;
figure
contour3(exact)
```

3. Comparación de resultados con solución real.

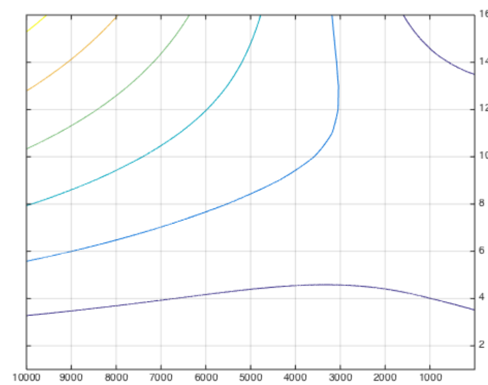
Para el siguiente ejemplo, se utilizaron los siguientes parámetros:

- $u(x,t) = e^{-3t}sin(x) + xt$
- $f(x,t) = x$
- $g(x) = sin(x)$
- $t \in [0,1]$ en intervalos de 0,0001.
- $x \in [0,\pi]$ en intervalos de $\frac{\pi}{15}$.
- $\lambda < 1$

3.1. Lineas de contorno: vista superior



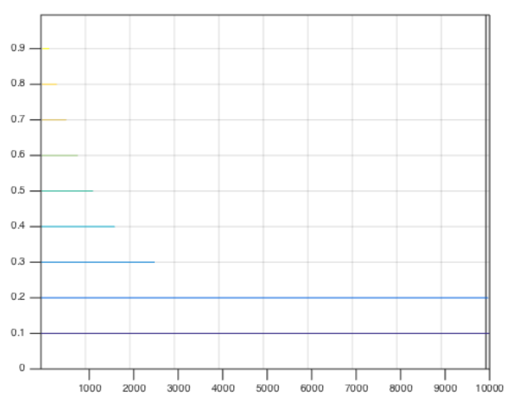
(a) Estimado



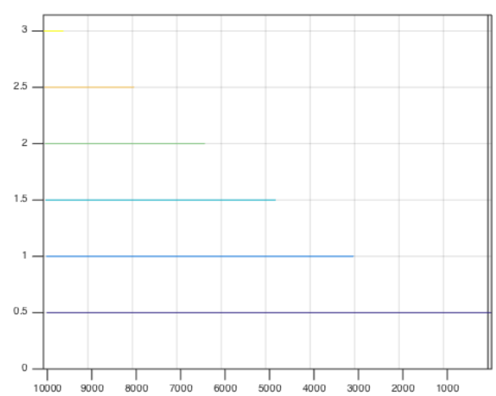
(b) Real

Aquí podemos apreciar claramente, que la estimación mediante diferencias finitas se asemeja a la solución real. Si bien, no son exactamente iguales, el comportamiento es bastante similar.

3.2. Lineas de contorno: vista lateral

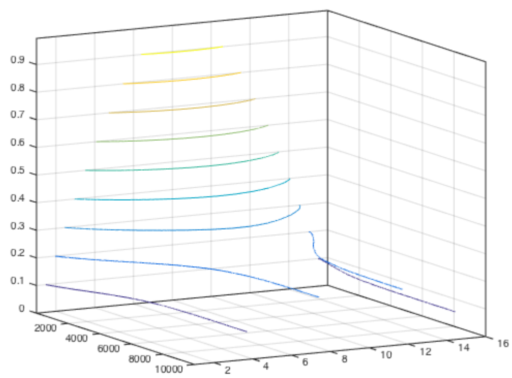


(a) Estimado

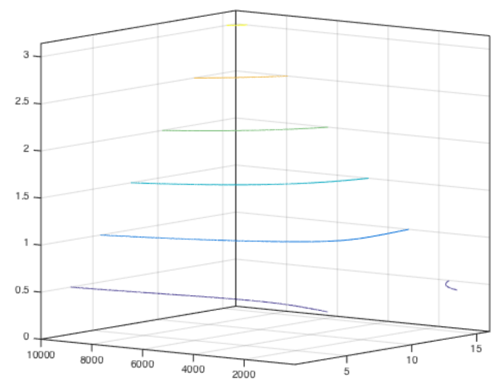


(b) Real

3.3. Lineas de contorno: vista diagonal



(a) Estimado



(b) Real

Por lo tanto, podemos apreciar gráficamente, que la estimación mediante diferencias finitas es una herramienta muy útil, que permite encontrar rapidamente estimaciones de funciones que pueden ser complejas de encontrar en primera instancia.

Las diferencias (error matemático), es una de las grandes desventajas de este método, el cual se arrastra y puede ser incremental, y es por eso que el parámetro λ debe ser menor a 1, para asegurar que el problema se comporte de manera *estable* (aunque esto no asegura que el problema converja totalmente).

4. Código fuente.

Todo el contenido de este trabajo se puede encontrar en `https://github.com/XzAeRo/Academic-Code/tree/master/Mat015`