Introducción al Modelado del Continuo

Trabajo Práctico $n°$ 3: Ecuación del calor.

Estabilidad

Se desea resolver el siguiente problema:

$$\begin{cases} u_t(x,t) = \alpha u_{xx}(x,t) & \text{en } \Omega \times [0,T_f] \\ u(0,t) = u(1,t) = 0 & \\ u(x,0) = g(x) & \end{cases},$$

donde $\Omega = [0,1]$.

Para ello, planteamos una grilla de Ω con paso h y un paso temporal Δt y proponemos dos métodos: el **explícito**, cuya discretización es:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \delta_x^2(u_i^n),$$

donde

$$\delta_x^2(u_i^n) = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{h^2}.$$

Y el **implícito**, en el que el operador de derivada segunda se aplica sobre u_i^{n+1}.

Implementar funciones que apliquen cada uno de estos métodos y probarlas para distintos valores de $r = \alpha\frac{\Delta t}{h^2}$ en torno a $r = \frac{1}{2}$. Realizar animaciones del resultado obtenido. Indicar qué se observa. En casos en los que el método se mantenga estable, variar el valor de α y comentar qué efecto produce esta variación.

Utilizar un dato inicial g que satisfaga las condiciones de contorno.

Velocidad de ejecución

Consideramos ahora el problema bidimensional, donde u_{xx} es reemplazado por Δu, $\Omega = [0,1] \times [0,1]$ y las condiciones de contorno de tipo Dirichlet homogéneas se aplican a todo el borde. Trabajaremos sólo con el método **implícito**. Compararemos la performance al utilizar matrices llenas o matrices ralas. Para construir matrices llenas pueden resultar útiles las siguientes funciones de la librería `LinearAlgebra`: `diagm` recibe las diagonales a cargar en una matriz; `Symmetric` crea una matriz simétrica a partir de su parte triangular superior. El operador `\` resuelve un sistema (`A\b`). Altenativamente las matrices pueden crearse como matrices *ralas*, implementadas en la librería `SparseArrays`. `SparseArrays` exporta, entre otras, las funciones `sparse` (que convierte en rala una matriz llena), `spdiagm`, similar a `diagm`, `blockdiag` que permite generar una matriz poniendo matrices más pequeñas a lo largo de su diagonal y `spzeros`, similar a `zeros`.

Además, el operador `\` puede aplicarse tanto a matrices como a descomposiciones. Más precisamente: supongamos que debemos resolver muchos sistemas con la misma matriz `A`. Podemos usar `A \ ...` cada vez, o podemos computar una descomposición apropiada de `A` (`lu` o `qr`, por ejemplo): `descA = lu(A)` y luego utilizar `\` sobre la descomposición: `descA \ b`. De esta manera la operatoria para descomponer la matriz se realiza una única vez y se almacena en una variable.

Utilizando `@benchmark` de la librería `BenchmarkTools` comparar la performance y la ocupación de memoria de rutinas que apliquen el método implícito con matrices ralas o llenas, con o sin precálculo de alguna descomposición adecuada.

Con la versión que resulte más eficiente, realizar una animación de evolución de la temperatura utilizando `surface` o `heatmap`.

Difusión con transporte

Por último, planteamos el siguiente problema, un poco más interesante, con $\Omega = [0, 1] \times [0, 1]$:

$$\begin{cases} u_t(x, y, t) &= \alpha \Delta u(x, y, t) + \beta u_x(x, y, t) \quad \text{en } \Omega \times \left[0, T_f\right] \\ u_y(x, 0, t) = u_y(x, 1, t) = 0 \\ u(0, y, t) &= u(1, y, t) \\ u(x, y, 0) &= g(x, y) \end{cases}.$$

Observar que se tienen condiciones de tipo Neumann homogéneas en las paredes horizontales y condiciones periódicas en las paredes verticales.

Escribir la matriz del sistema discreto, resolver y animar la solución. Utilizar valores pequeños de α y probar algunos valores de β, incluyendo negativos y positivos.

Utilizar como dato inicial la característica de una bola de radio $\frac{1}{4}$ centrada en $\left(\frac{1}{2}, \frac{1}{2}\right)$.
