FÍSICA COMPUTACIONAL: BLOQUE 2

Juan José Vidal Justamante

Diciembre 2021

1. Introducción

En este informe voy a comentar el procedimiento de forma breve en cada uno de los ejercicios propuestos para resolver. **Aquí también están hechos varios extras.** También habrá algún comentario acerca del código, aunque estos también tienen bastantes comentarios dentro.
Para ejecutar la parte correspondiente en cada código, basta con quitar las comillas del apartado.

2. Ecuaciones elípticas

Las ecuaciones elípticas tienen la forma:

$$\nabla^2\Phi(q_i) = f(q_i) \tag{1}$$

La ecuación de Poisson-Laplace es un buen ejemplo de elíptica. En esta parte del informe, resolveremos esta ecuación en concreto para encontrar potenciales electromagnéticos de distintas formas.
El ejemplo que resolveremos es el de una malla cuadrada con potencial 0 en todos los puntos excepto en uno de sus lados, el cual tendrá potencial $100V$. Lo vamos a plantear de 3 formas diferentes:

2.1. Breve explicación teórica de cada método

2.1.1. Método de Jacobi

A partir de las condiciones iniciales, es decir, el valor de potencial en cada punto, creamos un bucle que vaya recorriendo esta matriz de potenciales y, en una matriz nueva, vaya metiendo valores de este valor en el paso actual de tiempo. Esto se hace con la fórmula:

$$\phi'_{ij} = \frac{1}{4}\left(\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}\right) \tag{2}$$

Haremos que este bucle corra hasta que el valor actual del potencial y el anterior difieran menos que un error seleccionado. Así veremos cómo el sistema se aproxima al equilibrio.

2.1.2. Método de Gauss-Seidel

Este método sigue el mismo procedimiento que el método de Jacobi, pero en vez de colocar los valores del potencial en una matriz nueva, va sobreescribiendo la matriz antigua. Esto supone un gran avance en la rapidez en la que funciona el programa y se nota a la hora de sacar los resultados .

2.1.3. Método directo

Para resolver el problema de esta forma, debemos plantear el sistema de ecuaciones para cada uno de los puntos de la malla y utilizar libreriías que lo resuelvan. La dificultad está en saber plantear adecuadamente las matrices del sistema.

2.2. El código y los resultados

El código es de implementación sencilla en el caso de Jacobi y Gauss-Seidel, pero un poco tedioso con el método directo.
Al ejecutar la primera parte uno se da cuenta de la diferencia de tiempo entre los dos primeros métodos, incluso al estar ambos relajados.
La segunda parte del código es en gran parte la construcción de las matrices, y luego resolvemos en 2 líneas las ecuaciones matriciales.
En el código 1 hay comentarios en cada paso por si no queda claro.
Los resultados obtenidos son:

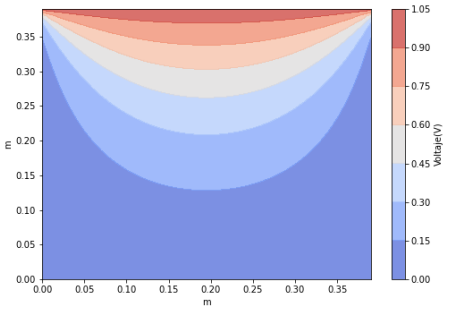

Figura 1: Ejemplo 1

También podemos probar con otras distribuciones de potencial. Si relajamos la condición del error, podemos aplicar Gauss Seidel (por ejemplo) a esta distribución:

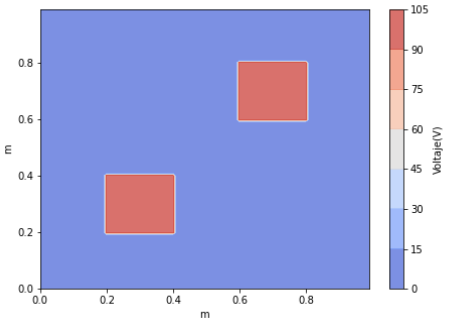

Figura 2: Condiciones iniciales

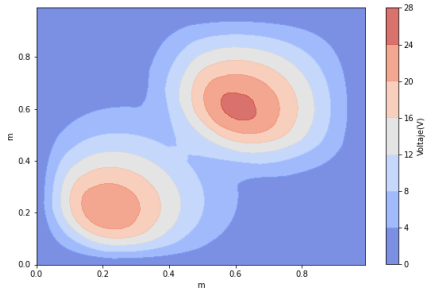

Figura 3: Situación final

3. Ecuaciones parabólicas

Las ecuaciones parabólicas son las que tienen la forma:

$$\frac{\partial \phi}{\partial t} = D\nabla^2 \phi \tag{3}$$

Donde D es una constante.
Los ejemplos más conocidos de este tipo de ecuación son la ecuación del calor y la ecuación de Schrödinger. Vamos a ver 2 métodos que harán sencilla la resolución de esta ecuación que a priori parece compleja.

3.1. Breve explicación teórica de cada método

3.1.1. Método FTCS

En resumen, este método discretiza las dos derivadas y despeja para obtener el valor de cada paso del tiempo en función de distintas posiciones en el tiempo anterior. La ecuación final es:

$$\phi_i^{k+1} = \phi_i^k + D\frac{\Delta t}{(\Delta x)^2}\left(\phi_{i+1}^k + \phi_{i-1}^k - 2\phi_i^k\right) \tag{4}$$

En esta ecuación los superíndices k muestran los pasos del tiempo y los subíndices i los espaciales.
Es importante notar que este es el caso para una dimensión. El proceso para obtener esta ecuación en más dimensiones es análogo, simplemente hay que añadir términos espaciales al laplaciano (Ecuación 3) y despejar.

3.1.2. Crank-Nicholson

El método de Crank-Nicholson utiliza las diferencias *fordward* y *backward* de Euler para llegar a una expresión similar a la del método FTCS. En lo que se diferencia es que no sólo hay un término de la función en el instante presente, aparecerán más.

$$-ru_{i+1}^{k+1} + (1+2r)u_i^{k+1} - ru_{i-1}^{k+1} = ru_{i+1}^k + (1-2r)u_i^k + ru_{i-1}^k \tag{5}$$

$$r = \frac{D\Delta t}{2(\Delta x)^2} \tag{6}$$

Esto se soluciona planteando la ecuación en forma de dos matrices tridiagonales, con la parte de la izquierda de la ecuación asociada al instante actual y la derecha al anterior.
En todos los programas que comentaré a continuación, vamos a implementar una función a la que le pases las matrices de coeficientes asociadas a ambos lados de la ecuación 5 y las condiciones iniciales. Esta función hará los cálculos y devolverá los datos para cada uno de los instantes de tiempo, para poder hacer

las animaciones.

3.2. El código y los resultados

3.2.1. Método FTCS

Aquí hemos implementado este código para dos casos diferentes: en una dimensión y en 2.

-CASO 1D.
Tenemos una placa de acero que separa dos medios, uno de agua caliente a $50°C$ y otro de agua fría a $0°C$. Inicialmente se encuentra a $20°C$. Si aplicamos el método FTCS, debemos poder ver cómo con el paso del tiempo la temperatura pasa a estabilizarse en cada diferencial de longitud de la barra, pasando de una temperatura caliente máxima a una fría mínima en un cambio representado por una línea recta. Lo que hemos hecho ha sido plotear una animación para ver la temperatura de cada diferencial en cada paso de tiempo. Esta animación está disponible ejecutando el código *difusión1d.py*, en la carpeta *Parabólicas*.

-CASO 2D
Aquí vamos a resolver la ecuación del calor para un cuadrado con un círculo interior y en el centro con una temperatura superior. La temperatura dentro de este círculo es de $700°C$ y fuera de él $300°C$. En este caso será más conveniente y visual hacer una representación en 2D del instante final:
Aquí se aprecia claramente cómo la temperatura de los bordes del cuadrado es

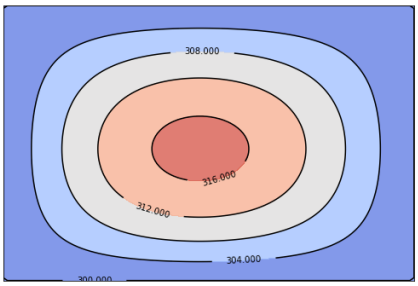

Figura 4: Esquema de temperaturas al aplicar FTCS

de $300°C$, ya que no ha experimentado aún un cambio. Por otra parte, el centro se ha ido enfriando y ha habido un flujo de calor desde el círculo condición inicial hacia fuera, aumentando la temperatura como se ve bien en cada superficie de calor.

3.2.2. Cranck-Nicholson

Este método lo vamos a utilizar para resolver 2 casos diferentes. Resolveremos otro caso de la ecuación de calor en 1D y la ecuación de Schrödinger para un pozo infinito.
-CASO 1.
El ejemplo propuesto es el de una barra unidimensional con los bordes puestos en contacto con un material a $0°C$. Nosotros esperaremos que esta barra, que está inicialmente a $100°C$, experimente un descenso de temperaturas desde los extremos, bajando de forma gradual hasta llegar al equilibrio, donde toda la barra está a $0°C$.
Este caso se resuelve con las matrices asociadas a la ecuación 5.
La animación está disponible al ejecutar el código *crank-nicholson.py*, en la carpeta *Parabólicas*.

-CASO 2. Ahora vamos a resolver la ecuación de Schrödinger en 1D. De la misma forma que antes, esperamos ver cómo está representada la función a cada paso del tiempo, y por ello vamos a hacer una animación. Además, al tratarse de un pozo infinito, asumimos que el potencial es 0 en la región que vamos a estudiar y así la ecuación queda análoga a la anterior.
El código es bastante simple, lo único que resulta difícil es implementar las condiciones de contorno de la onda. Teniendo ya la función crank-nicholson, simplemente basta con implementarla. La animación se puede ver en el código *schrodinger1d.py*, en la carpeta *Parabólicas*.

4. Ecuaciones hiperbólicas

Las ecuaciones hiperbólicas tienen la forma:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{7}$$

Es decir, este tipo de ecuaciones relaciona la diferencial segunda del tiempo con el laplaciano. La ecuación 7 es la ecuación de ondas, pero también se puede aplicar a la del telégrafo:

$$\frac{\partial^2 u}{\partial t^2} + \alpha \frac{\partial u}{\partial 2} + \beta u = c^2 \frac{\partial^2 u}{\partial x^2} \tag{8}$$

4.1. Breve explicación teórica de cada método

4.1.1. Método FDTD

Este método es de sencilla deducción: simplemente aplicamos diferencias centradas para ambas partes de la ecuación y despejamos para el término actual(u_i^{k+1}). Llegamos a la expresión siguiente:

$$u_i^{k+1} = \left(\frac{c^2\Delta t^2}{\Delta x^2}\right)\left(u_{i+1}^k + u_{i-1}^k - 2u_i^k\right) + \left(2u_i^k - u_i^{k-1}\right) \tag{9}$$

La parte más tediosa acerca de este método es la búsqueda de condiciones iniciales y de contorno. Se ve claramente que necesitaremos datos acerca de la función u dos pasos de tiempo por detrás, y por lo tanto necesitaremos más datos para inicializar el método.

4.1.2. Método implícito

Este método es el equivalente a Crank-Nicholson para este tipo de ecuaciones. Promediamos las partes espaciales y temporales y llegamos a la expresión:

$$u_i^{k+1} + u_i^{k-1} - 2u_i^k = \left(\frac{c^2\Delta t^2}{\Delta x^2}\right)\left(u_{i+1}^{k+1} + u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k-1} + u_{i-1}^{k-1} - 2u_i^{k-1}\right) \tag{10}$$

Esta expresión se puede descomponer en matrices tridiagonales de una forma análoga al caso de Crank-Nicholson. En este caso no quedará únicamente un producto, pero no se complica demasiado a la hora de resolverse.

4.2. El código y los resultados

4.2.1. Onda ejemplo

Vamos a ver cómo resolvemos el caso de una onda a la que se le aplica un pulso triangular y la soltamos en reposo.
Las condiciones iniciales para el estado inicial las conocemos, es decir, sabemos cómo soltamos la cuerda. Además, sabiendo que la soltamos en reposo, podemos aplicar Taylor y quedarnos en el primer orden. Con ello deducimos que la posició en el instante 1 será la misma que en el 0 aproximadamente, y podemos inicializar el programa *Hiperbólicas.py*, resuelto de dos formas: mediante FDTD y de forma implícita, ambos en el programa. Aquí encontraremos la animación mostrada dos veces.

4.2.2. Ondas amortiguadas

En este caso no vamos a aplicar un método diferente, lo que ocurre es que a la hora de usar cualquiera de los métodos comentados debemos volver a deducir las expresiones teniendo en cuenta algún término adicional. Si le añadimos un término a la ecuación de onda:

$$\frac{\partial^2 u}{\partial t^2} + \frac{2\kappa}{\rho}\frac{\partial u}{\partial 2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{11}$$

Aplicando el método FDTD:

$$u_i^{k+1} = \frac{1}{\nu}\left(\frac{u_{i+1}^k + u_{i-1}^k - 2u_i^k}{\Delta x^2} - \frac{u_i^k 2\kappa}{\Delta t \rho} + \frac{2u_i^k - u_i^{k-1}}{\Delta t^2}\right) \tag{12}$$

Donde:

$$\nu = \left(\frac{1}{\Delta t^2} + \frac{2\kappa}{\Delta t \rho}\right) \tag{13}$$

Aplicando este método de la misma forma que anteriormente, llegamos a una animación que cambia ligeramente el movimiento de la onda. Esta animación también está en el código *Hiperbólicas.py*.

4.2.3. Ecuación del telégrafo

Este ejemplo lo vamos a resolver aplicando FDTD a la ecuación planteada:

$$\frac{\partial^2 u}{\partial t^2} + \frac{\partial u}{\partial 2} + 2u = \frac{\partial^2 u}{\partial x^2} \tag{14}$$

Al aplicar el método llegamos a la expresión:

$$u_i^{k+1} = \frac{1}{\nu}\left(\frac{u_{i+1}^k + u_{i-1}^k - 2u_i^k}{\Delta x^2} - 2u_i^k + \frac{u_i^k}{\Delta t} + \frac{2u_i^k - u_i^{k-1}}{\Delta t^2}\right) \tag{15}$$

Donde:

$$\nu = \left(\frac{1}{\Delta t^2} + \frac{1}{\Delta t}\right) \tag{16}$$

Aplicando el método con las condiciones de contorno planteadas llegamos a la última animación del programa *Hiperbólicas.py*.

5. Ecuaciones de hidrodinámica

En esta última parte utilzaremos dos métodos para resolver los siguientes tipos de ecuaciones:

Ecuación de advección:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \tag{17}$$

Y la ecuación de Burgers:

$$\frac{\partial u}{\partial t} + \epsilon u \frac{\partial u}{\partial x} = 0 \tag{18}$$

5.1. Método Lax-Wendroff

El método Lax-Wendroff consiste en aplicar Taylor con diferencias finitas sobre la ecuación de advección hasta llegar a la expresión:

$$u_i^{k+1} = u_i^k - \alpha(u_{i+1}^k - u_{i-1}^k) + \beta(u_{i+1}^k - 2u_i^k + u_{i-1}^k) \tag{19}$$

Con $\alpha = c\Delta t/2\Delta x$ y $\beta = c^2\Delta t^2/2\Delta x^2$ Este método lo usamos para resolver el ejemplo propuesto en clase. La animación está en la primera parte del programa *Hidrodinámica.py*.

5.2. Burgers

Para resolver la ecuación, este método hace un proceso similar al anterior, con la ecuación final:

$$u_i^{k+1} = u_i^k - \frac{\beta}{4}((u_{i+1}^k)^2 - (u_{i-1}^k)^2) +$$
$$\frac{\beta^2}{8}((u_{i+1}^k + u_i^k)((u_{i+1}^k)^2 - (u_i^k)^2) - (u_i^k + u_{i-1}^k)((u_i^k)^2 - (u_{i-1}^k)^2))$$

Con este método hemos resuelto el segundo ejemplo propuesto, la animación resultado está en la última parte del programa *Hidrodinámica.py*.