Informe 05 - Resolución numérica de la ecuación de Poisson 2D

Profesor: Valentino González
Auxiliar: Felipe Pesce
Integrantes: Nicolás Troncoso[1]
[1] *Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Física.*

I. INTRODUCCIÓN

En el siguiente informe se presenta la resolución de dos ecuaciones en derivadas parciales (en adelante EDPs) utilizando en conjunto los algoritmos de Crank-Nicholson y Euler.

La motivación principal de este trabajo es encontrar un método que permita resolver ecuaciones de la forma:

$$\frac{\partial m}{\partial t} \;=\; \gamma \frac{\partial^2 m}{\partial x^2} \;+\; f(m), \tag{I.1}$$

donde $m = m(x,t)$ y $f(m)$ son funciones y γ es una constante. Se tienen condiciones iniciales y de borde dadas. Este problema representa un desafío pues requiere combinar las soluciones de dos algoritmos por separado para solucionar este problema en conjunto.

En la sección **II** se presenta el algoritmo a utilizar y su implementación. En la sección **III** se presentan dos casos específicos donde será utilizado. En la sección **IV** se muestran los resultados obtenidos. En la sección **V** se discutirán los resultados y se concluirá sobre ellos.

II. ALGORITMO

Tenemos el método de Crank-Nicolson que se utiliza para resolver problemas de la forma:

$$\frac{\partial m}{\partial t} \;=\; \gamma \frac{\partial^2 m}{\partial x^2}. \tag{II.1}$$

El algoritmo discretiza las coordenadas espaciales y temporales de la forma:

$$m_j^n \;=\; m(t = t_n,\; x = x_j), \tag{II.2}$$

donde $t_n = t_0 + n\Delta t$ y $x_i = x_0 + i\Delta x$, con Δt y Δx los espaciados temporales y espaciales respectivamente. Si escribimos el método de Crank-Nicolson en su versión discretizada obtenemos:

$$\frac{m_i^{n+1} - m_i^n}{\Delta t} \;=\; \frac{\gamma}{2}\left(\frac{m_{i+1}^n - 2m_i^n + m_{i-1}^n}{(\Delta x)^2} \;+\; \frac{m_{i+1}^{n+1} - 2m_i^{n+1} + m_{i-1}^{n+1}}{(\Delta x)^2} \right). \tag{II.3}$$

Los problemas a resolver son más generales que los contemplados en la ecuación (II.1), en efecto, debemos agregar la funcion $f(n)$ discretizada a nuestras ecuaciones:

$$\frac{m_i^{n+1} - m_i^n}{\Delta t} \;=\; \frac{\gamma}{2}\left(\frac{m_{i+1}^n - 2m_i^n + m_{i-1}^n}{(\Delta x)^2} \;+\; \frac{m_{i+1}^{n+1} - 2m_i^{n+1} + m_{i-1}^{n+1}}{(\Delta x)^2} \right) \;+\; f(m_i^n). \tag{II.4}$$

Definimos la variable η como:

$$\eta \;=\; \gamma \frac{\Delta t}{2(\Delta x)^2}, \tag{II.5}$$

y obtenemos:

$$-\eta m_{i+1}^{n+1} \;+\; (1+2\eta) m_i^{n+1} \;-\; \eta m_{i-1}^{n+1} \;=\; \eta(m_{i+1}^n \;-\; 2m_i^n \;+\; m_{i-1}^n) \;+\; \Delta t f(m_i^n) + m_i^n. \tag{II.6}$$

Ordenando los términos y expresandolos en su forma matricial:

$$\begin{bmatrix} (1+2\eta) & -\eta & & 0 \\ -\eta & \ddots & \ddots & \\ & \ddots & \ddots & -\eta \\ 0 & & -\eta & (1+2\eta) \end{bmatrix} \begin{bmatrix} m_1^{n+1} \\ \vdots \\ \vdots \\ m_{if}^{n+1} \end{bmatrix} = \begin{bmatrix} \eta(m_2^n - 2m_1^n) + m_1^n + \Delta t f(m_1^n) \\ \vdots \\ \vdots \\ \eta(-2m_{if}^n + m_{if-1}^n) + m_{if}^n + \Delta t f(m_{if}^n) \end{bmatrix} \tag{II.7}$$

donde se considero if como el paso final en la discretización espacial. Dado que este problema es del tipo $Ax = b$ con A una matriz y b un vector conocido (en efecto, basta iterar a partir de las condiciones iniciales para conocer cualquier tiempo t_{n+1}), basta aplicar un algoritmo de resolución de sistemas de ecuaciones lineales para resolverlo. Para este caso en específico se utilizó `numpy.linalg.solve`(A,b) [1] que toma la matriz A y el vector b y devuelve el vector x.

Las condiciones de borde especificas para cada problema fueron introducidas como modificaciones en A y b de manera que se obligase a m_1^n y m_{if}^n a tomar valores específicos.

III. CASOS A ESTUDIAR

A. Fisher-KPP

El primer caso a estudiar consiste en la ecuación de Fisher-KPP en 1D, y que busca modelar el comportamiento animal.

$$\frac{\partial m}{\partial t} = \gamma \frac{\partial^2 m}{\partial x^2} + \mu\, m\,(1 - m). \tag{III.1}$$

Los puntos de equilibrio se obtienen de analizar la ecuación:

$$\frac{\partial m}{\partial t} = \mu\, m\,(1 - m), \tag{III.2}$$

en efecto, un punto fijo tiene derivada espacial nula. Podemos ver que la derivada temporal de m se anula para $m = 0$ y $m = 1$. Si derivamos nuevamente obtenemos:

$$\frac{\partial^2 m}{\partial t^2} = \mu\,(1 - 2m), \tag{III.3}$$

y reemplazando los puntos de equilibrio vemos que $m = 1$ es equilibrio estable, mientras que $m = 0$ es equilibrio inestable. El sistema fue resuelto utilizando el método de Crank-Nicolson modificado y ocupando $x \in [0, 1]$ y $t \in [0, 4]$. Las condiciones de borde e iniciales fueron:

$$\begin{aligned} m(t, 0) &= 1 \\ m(t, 1) &= 0 \\ m(0, x) &= e^{-x^2/0{,}1} \end{aligned} \tag{III.4}$$

Se ocupó una discretización de 500 pasos en espacio y 500 pasos en tiempo y se utilizarón las constantes $\gamma = 0{,}001$ y $\mu = 1{,}5$, todo en sistema adimensional.

B. Newell-Whitehead-Segel

La ecuación de Newell-Whitehead-Segel busca modelar fenómenos de combustión, convección, magnetización, entre otros, y está dada por:

$$\frac{\partial m}{\partial t} = \gamma \frac{\partial^2 m}{\partial x^2} + \mu\, m\,(1 - m^2). \tag{III.5}$$

Los puntos de equilibrio se obtienen de analizar la ecuación:

$$\frac{\partial m}{\partial t} = \mu\, m\,(1 - m^2), \tag{III.6}$$

Podemos ver que la derivada temporal de m se anula para $m = -1$, $m = 0$ y $m = 1$. Si derivamos nuevamente obtenemos:

$$\frac{\partial^2 m}{\partial t^2} = \mu\,(1 - 3m^2), \tag{III.7}$$

y reemplazando los puntos de equilibrio vemos que $m = 1$ es equilibrio estable al igual que $m = -1$, mientras que $m = 0$ es equilibrio inestable. El sistema fue resuelto utilizando el método de Crank-Nicolson modificado y ocupando $x \in [0, 1]$ y $t \in [0, 4]$. Las condiciones de borde e iniciales fueron:

$$\begin{aligned} m(t, 0) &= 0 \\ m(t, 1) &= 0 \\ m(0, x) &= \texttt{random(-0.3, 0.3)} \end{aligned} \tag{III.8}$$

En efecto, se utilizó como condición inicial un número random en el intervalo [−0,3, 0,3], para ello se utilizó la función `numpy.random.uniform` perteneciente a la biblioteca `random` de `numpy`. Se ocupó una discretización de 500 pasos en espacio y 500 pasos en tiempo y se utilizarón las constantes $\gamma = 0{,}001$ y $\mu = 1{,}5$, todo en sistema adimensional.

IV. RESULTADOS

A. Fisher-KPP

Para la ecuación de Fisher-KPP se obtuvieron los siguientes resultados:

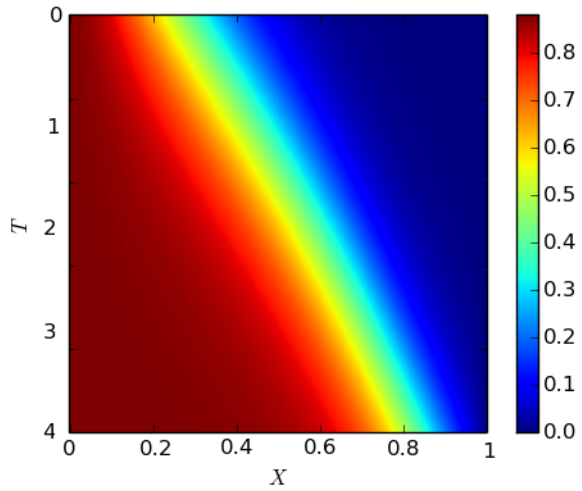

Figura 1: Diagrama de posición X, tiempo T y densidad m en escala de colores, todas en sistema normalizado de unidades. Los colores están exagerados utilizando $arcsinh(m)$.

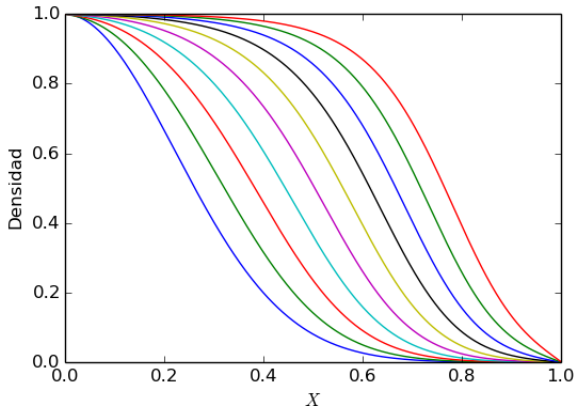

Figura 2: Densidad m de la especie en el eje X. Las distintas curvas son para diferentes tiempos, podemos ver que a medida que avanza el tiempo la curva se va trasladando hacia mayores x.

B. Newell-Whitehead-Segel

Para la ecuación de Fisher-KPP se obtuvieron los siguientes resultados:

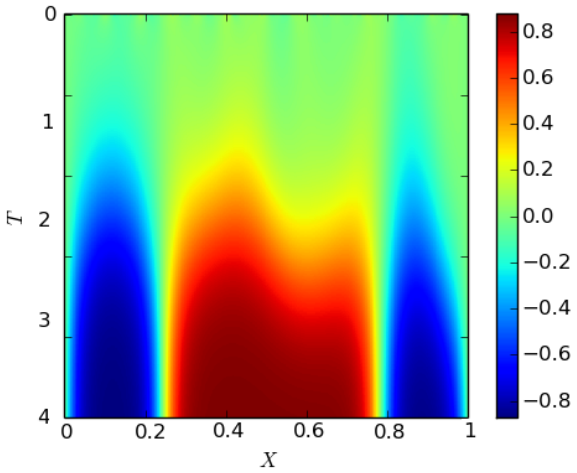

Figura 3: Diagrama de posición X, tiempo T y densidad m en escala de colores, todas en sistema normalizado de unidades. Los colores están exagerados utilizando $arcsinh(m)$.

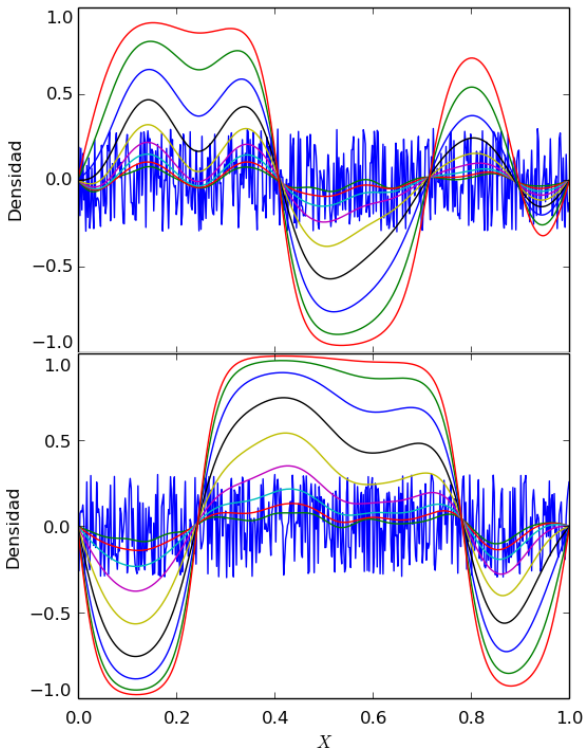

Figura 4: Densidad m en el eje X. Las distintas curvas son para diferentes tiempos, podemos ver que a medida que avanza el tiempo la curva se va trasladando hacia los puntos estables de equilibrio. Para el resultado superior se utilizó `numpy.random.seed`(251093), mientras que para el inferior se utilizó `numpy.random.seed`(98103).

V. DISCUSIÓN Y CONCLUSIÓN

Podemos ver que el algoritmo modificado de Crank-Nicolson es una buena estrategia para resolver EDPs de la forma de la ecuación (I.1), que es bastante más general que las ecuaciones tipo calor que resuelve Crank-Nicolson original.

Para la ecuación de Fisher-KPP obtuvimos los resultados esperados: La especie comienza a colonizar el espacio disponible, mientras que para la ecuación Newell-Whitehead-Segel se comprueba la inestabilidad de la densidad 0 y vemos que el sistema tiende a los puntos -1 o 1. La repetición de este algoritmo utilizando otras semillas para la randomización entrega el mismo tipo de resultados, como podemos ver en la figura 4.

Aunque existen maneras alternativas de resolver problemas de este tipo a través de Crank-Nicolson-Euler que son más explícitas en sus términos, la implementación utilizada en este informe resulta más rápida y sencilla de programar en tiempo persona. Queda pendiente para un siguiente trabajo analizar la diferencia en tiempo de computador entre el método explícito y el método de resolución $Ax = b$.

---

[1] `http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.solve.html`