Métodos numéricos para la Ciencia e Ingeniería
Tarea 6: Ecuaciones de reacción-difusión

Felipe Toledo Bittner

November 6, 2015

1 Introducción

Los sistemas de reacción-difusión se utilizan para modelar matemáticamente fenómenos espaciales en que el valor de una variable en cada punto presenta dependencia tanto de reacciones locales como de difusión desde los puntos cercanos. Su forma general puede verse en la ecuación (1). El primer término de la derecha corresponde a la componente difusiva y el segundo a la reactiva, en este caso modelada como el polinomio $P(n)$.

$$\frac{dn}{dt} = \gamma \frac{d^2 n}{dx^2} + P(n) \tag{1}$$

En este trabajo se implementa una clase en python que permite crear un sistema de reacción-difusión unidimensional y resolverlo para un intervalo de tiempo escogido por el usuario. Se incluyen métodos que permiten setear la variable γ, los coeficientes del polinomio $P(n)$ y las condiciones iniciales y de borde.

2 Descripción de la solución

La clase implementada permite resolver cualquier sistema de reacción-difusión que posea la forma de la ecuación (1). En el archivo *main_p1.py* adjunto a este informe hay un ejemplo bien comentado con los pasos a seguir para inicializar el programa, en este caso con la ecuación de Fisher-KPP.

2.1 Algoritmo de integración

Para integrar la ecuación de forma numérica se opta por utilizar dos métodos, Crank-Nicolson para la parte de difusión y una versión modificada del método de Euler para la de reacción. La modificación ha sido una elaboración propia, y es justificada a continuación.
Inicialmente consideremos el caso en que no hay componente de difusión. La ecuación (1) queda escrita en forma numérica como se muestra en (2). Los subíndices (x, t) en este caso representan posiciones y tiempos discretos y h es el paso temporal de la integración.

$$\frac{n_{x,t+1} - n_{x,t}}{h} = P(n_{x,y}) \tag{2}$$

Que se puede integrar usando el método de Euler, como se indica en la ecuación (3). Se observa que el lado derecho consiste simplemente en la actualización de la variable n usando su pendiente en cada instante de tiempo. Apartemos esto un momento para estudiar la parte difusiva, pero no debemos olvidarlo aún.

$$n^e_{x,t+1} = n_{x,t} + hP(n_{x,t}) \tag{3}$$

Suponiendo ahora que sólo hay componente difusiva, la ecuación original puede escribirse como (4). Aquí se utiliza el promedio entre las segundas derivadas calculadas hacia adelante y atrás para aproximar mejor el cambio en n.

$$\frac{n_{x,t+1} - n_{x,t}}{h} = \frac{1}{2}\left(\frac{n_{x+1,t+1} - 2n_{x,t+1} + n_{x-1,t+1}}{h^2} + \frac{n_{x+1,t} - 2n_{x,t} + n_{x-1,t}}{h^2}\right) \tag{4}$$

A partir de aquí se puede plantear un sistema de ecuaciones lineales para encontrar el valor de los $n_{x,t+1}$ desconocidos a partir de los $n_{x,t}$ conocidos para todos los puntos x simultáneamente. Éste es el llamado método de Crank-Nicolson, explicado en múltiples excelentes referencias [1].
El resultado de esta operación calcula todos los $n_{x,t+1}$. A grandes rasgos el resultado de una iteración puede describirse como se ve en la ecuación (5), donde $CN(\Delta x, h, \mathbf{n})$ es el término de actualización, Δx la distancia entre puntos en unidades de longitud físicas y $\mathbf{n}$ el vector que contiene todos los $n_{x,t}$.

$$n^{cn}_{x,t+1} = n_{x,t} + CN(\Delta x, h, \mathbf{n}) \tag{5}$$

En principio, calcular el valor completo de $n_{x,t+1}$ debiese ser tan sencillo como sumar $n_{x,t+1} = n^{cn}_{x,t+1} + n^{e}_{x,t+1}$, pero al hacer esto aparece un problema. Como se ve en la ecuación (6), al hacer esta operación aparece un término $2n_{x,t}$. Este término es espurio y debe corregirse ya que el espíritu de la integración numérica es ir actualizando $n_{x,t}$ usando sus tasas de cambio, dadas en este caso por $CN(...)$ y $P(n)$.

$$n_{x,t+1} = n^{cn}_{x,t+1} + n^{e}_{x,t+1} = \mathbf{2n_{x,t}} + CN(\Delta x, h, \mathbf{n}) + hP(n_{x,t}) \tag{6}$$

Para corregir el inconveniente, se opta por modificar el método de euler, dejándolo de la forma $n^{e^*}_{x,t+1} = hP(n_{x,t})$. Tras esta modificación, la ecuación (6) toma la forma (7), que como se ve en secciones posteriores logra integrar correctamente el sistema de difusión-reacción.

$$n_{x,t+1} = n^{cn}_{x,t+1} + n^{e^*}_{x,t+1} = n_{x,t} + CN(\Delta x, h, \mathbf{n}) + hP(n_{x,t}) \tag{7}$$

3 Resultados

3.1 Caso 1

En este caso se modela un sistema de Fisher-KPP, el cual consiste en la siguiente ecuación:

$$\frac{\partial n}{\partial t} = \gamma \frac{\partial^2 n}{\partial x^2} + \mu n - \mu n^2 \tag{8}$$

En este caso n representa la densidad de una especie en función del tiempo y la posición. Los parámetros usados son:

1. Largo de 1 [m]
2. Discretizado de 500 puntos
3. $\gamma = 0.001$
4. $\mu = 1.5$
5. Resolución: $h = 0.001$[s], $\Delta x = 0.002$[m]

[1] Por ejemplo el documento "Solucion Numerica de la Ecuacion de Calor por el Metodo de las Diferencias Finitas", realizado por Miguel Caro C. del Departamento de Matematicas de la Universidad del Atlantico de Colombia, entre otros. Puede encontrarse en línea en el sitio http://www.icmc.usp.br/CMS/Arquivos/arquivos_enviados/BIBLIOTECA_113_RT_327.pdf

6. Condiciones de borde: $n(t, x = 0) = 1$ y $n(t, x = 1) = 0$
7. Condiciones iniciales: $n(0, x) = e^{-x^2/0.1}$
8. Tiempo final de integración: $t_f = 4[s]$

Tras integrar la ecuación para 4 segundos, se obtienen los resultados de la siguiente figura:

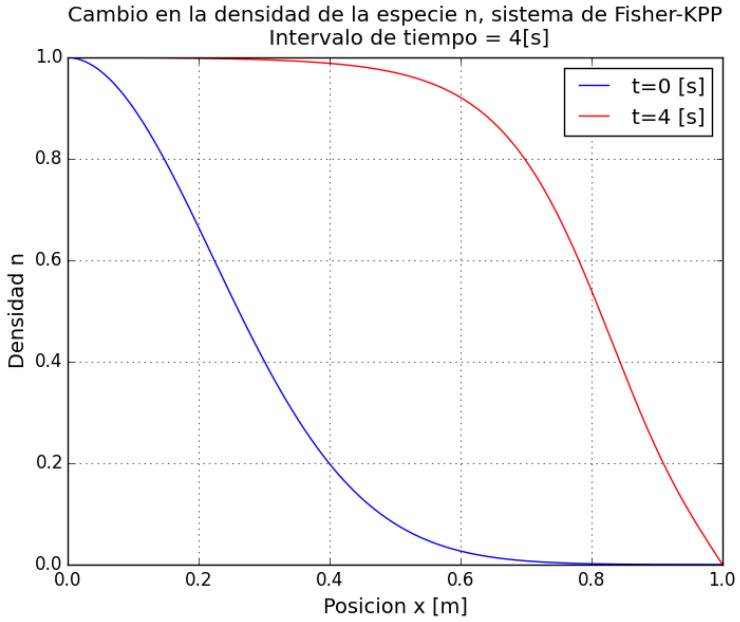

Figura 1: Resultados de la integración de la ecuación de Fisher-KPP.

Se observa que la densidad comienza a aumentar hacia la derecha con el paso del tiempo. Una forma de entender esto es viendo la condición de borde izquierda como una fuente de densidad, la que intenta difundir hacia la derecha generando una ola. A medida que nos alejamos de la fuente, comienza a dominar el término de reacción de competencia ($-\mu n^2$), acotando los máximos valores de n llegando a 0 en el extremo derecho debido a la otra condición de borde.

3.2 Caso 2

Ahora se integra la ecuación de Newell-Whitehead-Segel:

$$\frac{\partial n}{\partial t} = \gamma \frac{\partial^2 n}{\partial x^2} + \mu(n - n^3) \tag{9}$$

Usando las siguientes condiciones:

1. Largo de 1 [m]
2. Discretizado de 500 puntos
3. $\gamma = 0.001$
4. $\mu = 1.5$
5. Resolución: $h = 0.001$[s], $\Delta x = 0.002$[m]

6. Condiciones de borde: $n(t, x = 0) = 0$ y $n(t, x = 1) = 0$
7. Condiciones iniciales: $n(0, x) = random(-0.3, 0.3)$
8. Tiempo final de integración: $t_f = 4[s]$

En este caso se resuelve para dos conjuntos de condiciones iniciales donde cada n toma un valor aleatorio entre -0.3 y 0.3, generado usando la función *random.uniform()* de numpy. Con el fin de obtener resultados replicables se establecen dos semillas $s1 = 317$ y $s2 = 119$ con la función *random.seed(¡int¿)* de la misma librería.
Los resultados se presentan en la figura a continuación:

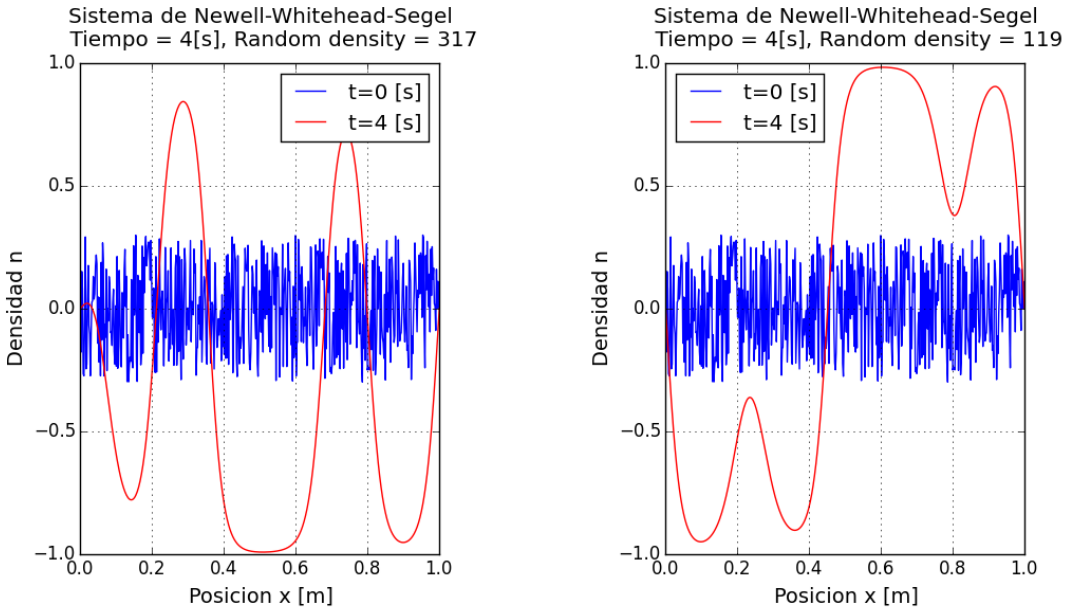

Figura 2: Resultados de la integración de la ecuación de Newell-Whitehead-Segel para dos condiciones iniciales distintas.

De las figuras se puede apreciar que la diferencia en las condiciones iniciales genera resultados completamente distintos tras el período de cálculo, lo que puede explicarse argumentando que se está integrando cerca de un punto de equilibrio inestable. El respaldo de esta afirmación proviene de que los valores aleatorios están acotados entre [-0.3, 0.3], por lo que se comienza a resolver desde puntos más cercanos a 0 (punto de equilibrio inestable) que a ± 1 (puntos de equilibrio estable).

4 Conclusiones

El principal logro de esta tarea es que se logra implementar código que permite trabajar con sistemas de difusión-reacción de forma general, operando además bajo el paradigma de programación orientada a objetos. Esto, junto a la documentación aquí presentada, los comentarios en el código y los ejemplos desarrollados debiese permitir a cualquier otra persona usar este código de forma explícita o encapsulada para resolver sus propios sistemas de difusión-reacción en una dimensión.
También se concluye que al utilizar varios métodos simultáneos de integración numérica se debe considerar su interacción para no obtener elementos redundantes en el cálculo, como se vió en la sección 2.1

Para el trabajo futuro se propone implementar optimizaciones de forma consciente en el código para disminuir el tiempo de cálculo de las rutinas más pesadas. Para este caso particular, por ejemplo, sería recomendable mejorar la velocidad del algoritmo de Crank-Nicolson, ya que es iterado muchas veces y toma un tiempo mayor que las otras rutinas debido a que realiza operaciones con matrices (inversión y multiplicación).