Métodos Numéricos para la Ciencia e Ingeniería
Informe Tarea 6

Benjamín Guerrero
18.862.370-0
3 de Noviembre, 2015

1. Pregunta 1

1.1. Introducción

En esta pregunta se pide resolver la ecuación de Fisher-KPP, que busca modelar el comportamiento de una especie animal. Esta es la siguiente:

$$\frac{\partial n}{\partial t} = \gamma \frac{\partial^2 n}{\partial x^2} + \mu n - \mu n^2 \quad (1)$$

En la cual la variable $n(t,x)$ describe la densidad de la especie en función del tiempo y la posición.

Esta ecuación tiene dos puntos de equilibrio en $n = 0$ (inestable) y $n = 1$ (estable). Las soluciones son una combinación entre una difusión y un pulso viajero.

Se pide usar el método de Crank-Nicolson para la difusión, y el método de Euler explícito para la reacción. Se debe usar x entre 0 y 1 con $\gamma = 0.001$ y $\mu = 1.5$, discretizando el espacio en 500 puntos aproximadamente, y considere las siguientes condiciones de borde:

$$n(t,0) = 1$$
$$n(t,1) = 0$$
$$n(0,x) = e^{-x^2/0.1}$$

Se debe elegir el paso temporal tal que la solución sea estable y se debe integrar hasta al menos $t = 4$.

Luego, se deben graficar los resultados.

1.2. Procedimiento

Se crea un archivo llamado FKPP.py, que se usará para hacer lo pedido. Primero, se importan*numpy* y *matplotlib.pyplot*.

Luego, se definen las condiciones de borde ya mencionadas con la función *definir_condiciones_borde*. Si bien esto se hace antes de definir el main, se sabe que la función se va a usar en el futuro, por lo que se programa ahora.

Ahora, después de definir γ, μ, y N_x (número de elementos de x, como 501) se redefine la longitud pedida de x, L_x como 1, y se define h (el paso en el espacio) como 0.002 (o sea, hay más o menos 500 pasos). Luego, se define $T = 10$ y el paso $dt = 0.1$. Así, el número de pasos temporales es $\frac{T}{dt} = 100$. Luego, se define el elemento auxiliar r como $r = \frac{\gamma * dt}{2h^2}$, para facilitar el cálculo de Crank-Nicolson (C-N).

Una vez hecho esto, se crean *np.arrays* con N_x ceros en su interior (*np.zeros)*, para los siguientes términos: $n_t, n_{tsig}, \alpha, \beta$ y b. También se define la matriz n_{sol}, que entregará el valor de n en el espacio para cada paso temporal (ayudará en los gráficos).

Se recuerda, que para una ecuación de difusión lineal de la forma:

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2}$$

Aplicar C-N, con:

$$r = \frac{a\Delta t}{(\Delta x)^2},$$

Resulta en:

$$-ru_{i+1}^{n+1} + 2(1+r)u_i^{n+1} - ru_{i-1}^{n+1} = ru_{i+1}^{n} + 2(1-r)u_i^{n} + ru_{i-1}^{n}$$

Primero, para atacar el problema, se ve que (1) es de la forma:

$$\frac{\partial n}{\partial t} = \gamma\left(\frac{\partial^2 n}{\partial x^2}\right) + \mu n - \mu n^2$$

Así, viendo lo anterior, la parte derecha, al aplicar C-N, se define como:

$$b_i = rn_{i+1}^k + rn_{i-1}^k + n_i^k(\mu * dt * \left(1 - n_i^k\right) + 1 - 2r)$$

(Las diferencias se deben a la definición de r)

De ahora en adelante, se usará i para la iteración en el espacio y k para iteración en el tiempo.

Ahora, como se vio en clases, realizar C-N es más fácil en términos de cálculo si se transforma el problema en uno de matriz tridiagonal. Realizando esto, se puede ver los valores α y β cumplen:

$$n_i = \alpha_i * n_{i+1} + \beta_i$$

Realizando algunos cálculos con la tridiagonal, también se puede sacar que:

$$\alpha_i = -\frac{A^+}{A^0 + A^- * \alpha_{i-1}}$$

$$\beta_i = \frac{b_i - A^- * \beta_{i-1}}{A^- * \alpha_{i-1} + A^0}$$

En este caso en particular, dadas las condiciones de borde y lo calculado:

$$A^+ = -r = A^-$$

$$A^0 = 1 + 2r$$

$$\alpha_0 = 0$$

$$\beta_0 = 1$$

Con este fin, se definen las funciones *calcular_b* y *calcular_alpha_y_beta*, que calculan lo pedido.

Luego, se define la función *avance_tiempo*, para que dé un paso en el tiempo, y que calcula n en ese paso (usando lo definido anteriormente, ($n_i = \alpha_i * n_{i+1} + \beta_i$)).

Con las funciones anteriores, ahora es posible iterar en el tiempo (en N_t) para calcular el valor de n en cada intervalo de tiempo. Estos valores de n se guardan en la primera fila de n_{sol}.

Ahora que tenemos los resultados, sólo queda graficarlos. Para esto, se usan distintos colores, para ver cómo evoluciona la función en el tiempo. También no se grafica cada resultado por intervalo de tiempo, sino que se grafica cada 10 intervalos (Cada t entero).

1.3. Resultados

Tras crear y correr el programa, el gráfico resultante es el que se puede ver en la página siguiente:

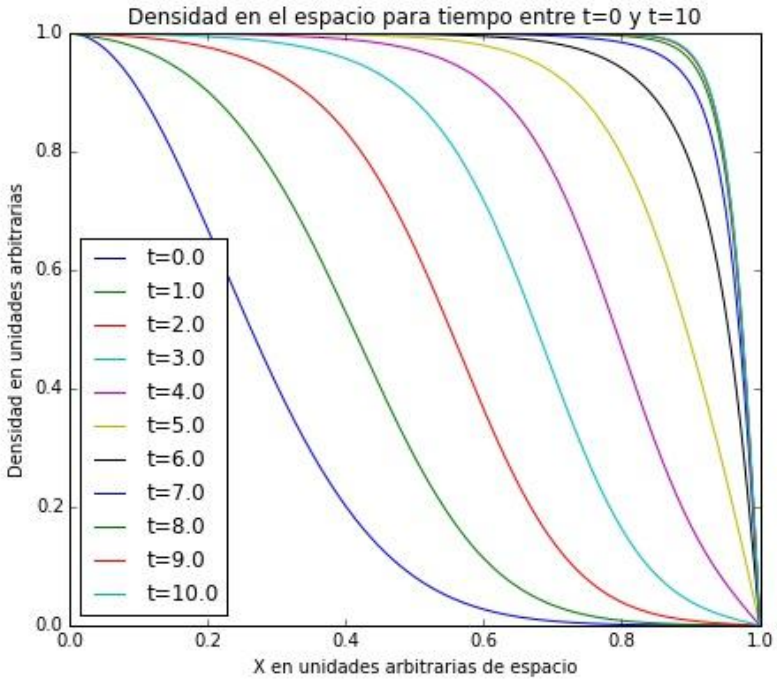

Fig 1: Grafico FKPP, para t entre 0 y 10.

2. Pregunta 2

2.1. Introducción

Ahora, se pide resolver la ecuación de Newell-Whitehead-Segel, la cual es la siguiente:

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

Se pide justificar porqué 1 y -1 son puntos de equilibrio estables para n, y se pide integrar la ecuación usando la misma estrategia que en la pregunta anterior (C-N), con las siguientes condiciones de borde:

$$n(t, 0) = 0$$
$$n(t, 1) = 0$$
$$n(0, x) = \texttt{np.random.uniform(low=-0.3, high=0.3, size=Nx)}$$

2.2. Procedimiento

Primero, se puede observar que esta ecuación es similar a la (1), de la pregunta anterior. La única diferencia es que, en lugar de un n^2, hay un n^3. En este caso, cuando n es 1 o -1, la ecuación queda:

$$\frac{\partial n}{\partial t} = \gamma\left(\frac{\partial^2 n}{\partial x^2}\right) + \mu * (0)$$

Por la parte anterior, sabemos que esto es un equilibrio estable en (1), por lo que podemos concluir que 1 y -1 son ptos de equilibrio estables en (2).

Ahora, creamos un archivo llamado *NWS.py*, que resolverá la ecuación. Debido a las similitudes con el programa anterior, sólo se van a señalar las diferencias. Lo que no se menciona a continuación es exactamente igual al programa anterior.

Primero, se usa *np.random.seed* para generar una semilla de valores aleatorios. Luego, en la función *definir_condiciones_borde*, en lugar de usar el exponencial de la parte anterior, x será un valor al azar entre -0.3 y 0.3, cuando t es 0. También cambia la primera condición de borde, lo que modifica los valores iniciales de los n calculados, y de β.

La función *calcular_b* se modifica para la nueva ecuación. Así:

$$b_i = rn_{i+1}^k + rn_{i-1}^k + n_i^k(\mu * dt * \left(1 - {n_i^k}^2\right) + 1 - 2r)$$

Finalmente, el gráfico es levemente modificado para que, esta vez, muestre los valores enteros pares del tiempo (0, 2, 4, 6, 8, 10).

2.3. Resultados

Se corre el programa 4 veces con distintas semillas. Los resultados son los siguientes:

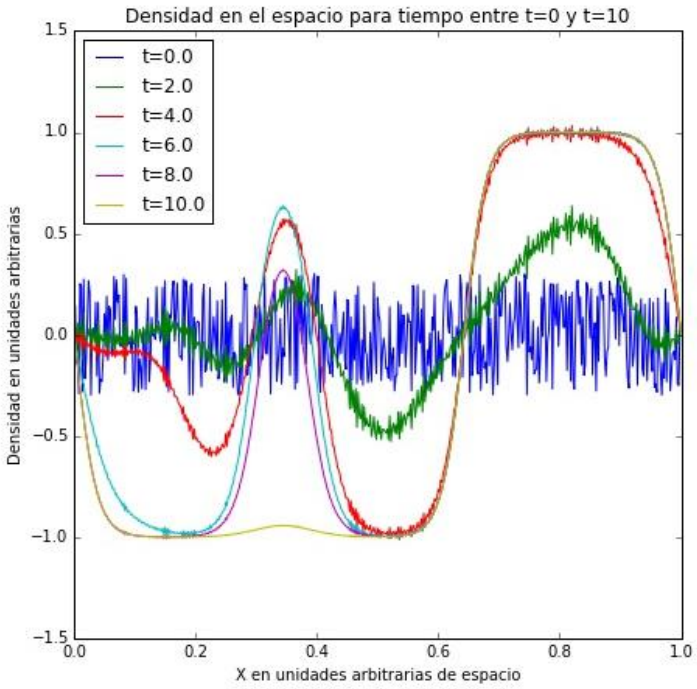

Fig 2: Gráfico NWS, con semilla=12.

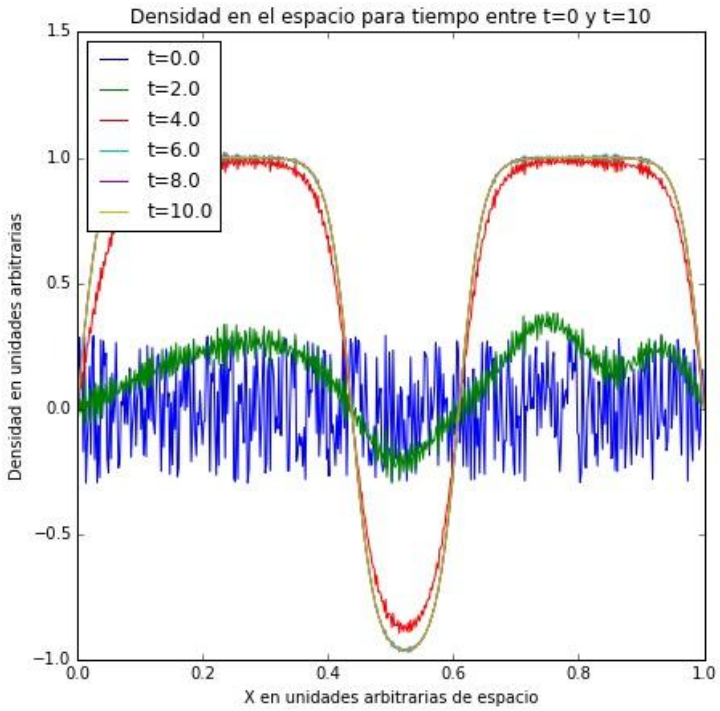

Fig 3: Gráfico NWS, con semilla=4.

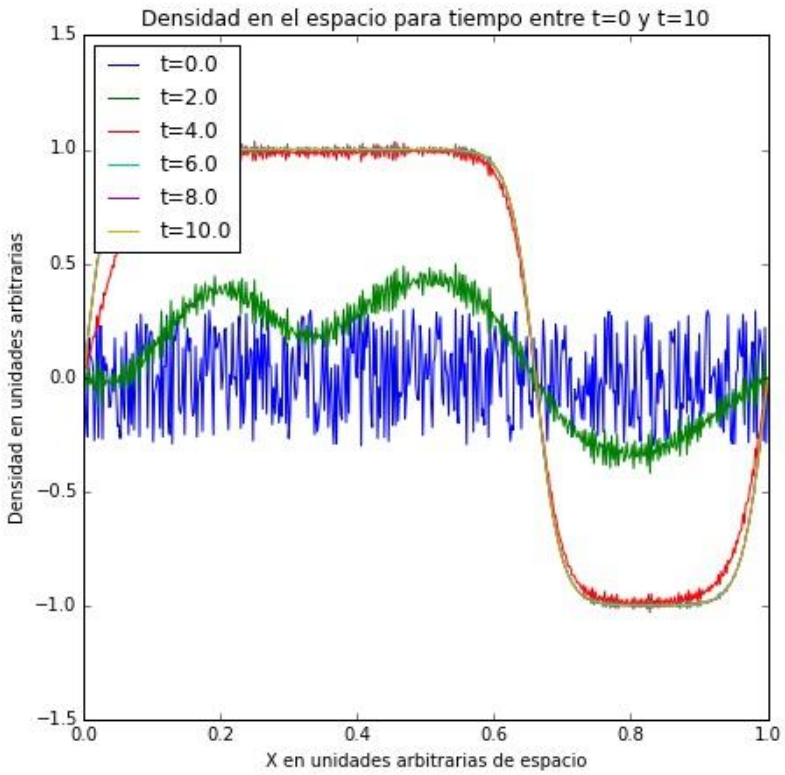

Fig 4: Gráfico NWS, con semilla=17

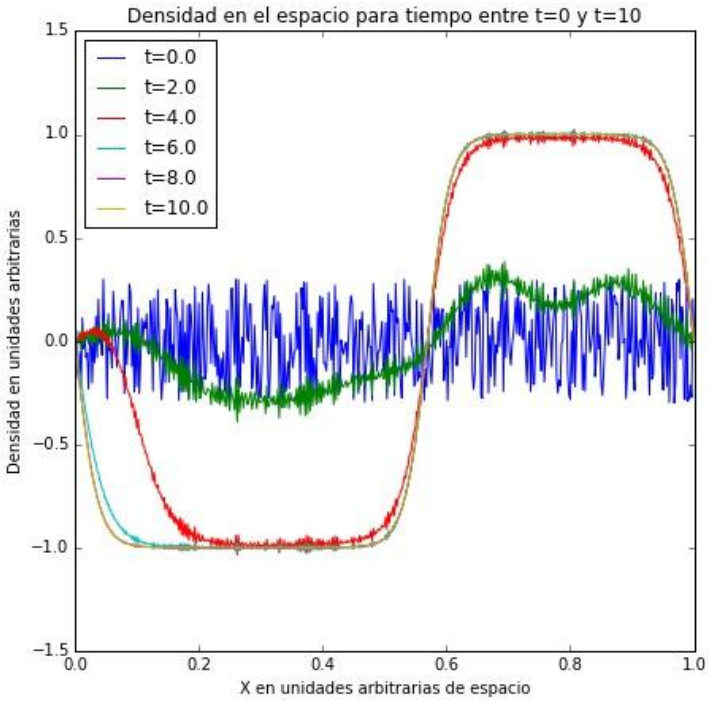

Fig 5: Gráfico NWS, con semilla = 25.

3. Conclusiones

Se puede observar que, en el gráfico 1, la densidad va a tender a ser 1 para todo x, solo siendo 0 en x=1 debido a su condición de borde definida. Esto es fácil de explicar debido a sus equilibrios.

En física, el equilibrio estable es la posición a la cual un objeto tiende a volver por efecto de la gravedad, si una fuerza externa lo aleja. Metafóricamente hablando, lo mismo pasa aquí. La condición inicial (la exponencial) fuerza a la densidad fuera de su equilibrio estable, pero esta vuelve a ese equilibrio con el pasar del tiempo. Además, se ve que la densidad tiende a alejarse de su equilibrio inestable.

Las figuras 2 a la 5 refuerzan lo dicho anteriormente. Sin importar el valor de la semilla (los valores al azar generados), n tiende a volver a sus equilibrios estables mientras más tiempo pasa, y a alejarse de sus equilibrios inestables.

También se concluye que el método de Crank-Nicolson es muy útil para este tipo de EDP, ya que se mantuvo estable con los valores dados, e incluso con las condiciones de borde al azar.