Modelo de la formación de radios en la aleta caudal del pez cebra

Para el estudio, y por tanto la modelización, de las formaciones de radios en la aleta caudal del pez cebra, es sabido que sigue un sistema de ecuaciones de derivadas parciales como el que sigue:

$$\frac{\partial x}{\partial t} = a(r - \overline{r})(R^2 - (r - \overline{r})^2 - b(j - \overline{j})) + \mu_1 \frac{\partial^2 r}{\partial x^2}$$

$$\frac{\partial j}{\partial t} = c(r - \overline{r}) - d(j - \overline{j}) + \mu_1 \frac{\partial^2 j}{\partial x^2}$$

donde:

- La unidad de la variable xt es la longitud característica de una célula
- La unidad de la variable t es el tiempo característico en el que se produce una división celular
- $r(x,t)$ es la concentración de una sustancia activadora de la formación de células de radio en x en el instante t, de forma que, una célula pasa a ser del tipo radio cuando el valor de r es mayor que un cierto valor crítico $\overline{r}$
- $j(t,x)$ es la concentración de una sustancia inhibidora de la formación de células de radio en x en el instante t
- $a, b, c, d, \overline{j}, R, \mu_1, \mu_2$ son constantes positivas

Se trata por tanto de un modelo de reacción-difusión: Las sustancias r y j son creadas y destruidas en las células y pasan de unas a otras a través de sus membranas por difusión.

```
# -*- coding: utf-8 -*-
from numpy import *
from matplotlib.pyplot import *
from mpl_toolkits.mplot3d import Axes3D

# Apartado c

def u0(x):
    return 1.0 * (10 <= x) * (x <= 20)

def v0(x):
    return -1.0 * (10 <= x) * (x <= 20)

def programa(L, T, Nx, u0, v0, delta, gamma, l):
    L = float(L)
    T = float(T)
    Nx = int(Nx)
    delta = float(delta)
    gamma = float(gamma)
    l = float(l)

    x = linspace(0, L, Nx + 1)  # partición del intervalo [0,L]
    dx = float(L / Nx)  # diferencia entre los distintos nodos
```

```
    dx2 = dx * dx

    # diferencia entre los distintos niveles de tiempo de manera que esté bajo la condicion CFL y así sea consistente y estable
    dt = min(0.39*dx2/(2*l), dx2/2)
    Mt = int(T / dt)

    # Los distintos valores para u^n (comenzando con los que proporciona u(x,0), es decir u0(x) "u^0")
    solu0 = u0(x)
    solv0 = v0(x)

    # Los distintos valores para u^(n+1)
    solu1 = 0 * solu0
    solv1 = 0 * solv0

    for n in range(1, Mt):
        tn = n * dt
        for i in range(1, Nx):

            # G(xi,tn) = solu0[i]*(1 - pow(solu0[i],2))-solv0[i]
            # H(xi,tn) = gamma * solu0[i] - delta * solv0[i]
            funG = solu0[i] * (1 - pow(solu0[i], 2)) - solv0[i]
            funH = gamma * solu0[i] - delta * solv0[i]
            solu1[i] = solu0[i] + dt * funG + dt / dx2 * (solu0[i - 1] - 2 * solu0[i] + solu0[i + 1])
            solv1[i] = solv0[i] + dt*funH + dt*l/dx2 * (solv0[i-1]-2*solv0[i] + solv0[i+1])

            # Condiciones frontera para u
            funG0 = solu0[0] * (1 - pow(solu0[0], 2)) - solv0[0]
            funGNx = solu0[Nx] * (1 - pow(solu0[Nx], 2)) - solv0[Nx]
            solu1[0] = solu0[0] + dt * funG0 + 2*dt/dx2*(solu0[1]-solu0[0])
            solu1[Nx] = solu0[Nx] + dt * funGNx + 2 * dt / dx2 * (solu0[Nx-1] - solu0[Nx])
            # Condiciones frontera para v
            funH0 = gamma * solu0[0] - delta * solv0[0]
            funHNx = gamma * solu0[Nx] - delta * solv0[Nx]
            solv1[0] = solv0[0] + dt*funH0 + 2*dt*l/dx2 * (solv0[1]-solv0[0])
            solv1[Nx] = solv0[Nx] + dt * funHNx + 2 * dt * l / dx2 * (solv0[Nx-1] - solv0[Nx])

        solu0 = solu1
        solv0 = solv1
    print('Para u: ' + str(solu0))
    print('Para v: ' + str(solv0))
    plot(x,solu0)
    plot(x,solv0)
    show()

# Apartado d
# Hemos tomado T = 10 y una partición del intervalo [0,L] en 70 nodos
programa(30,10,70,u0,v0,1.78,1.8,12)
```

```
71
```