UdeMM - Ingeniería de Sistemas
Simulación de modelos en derivadas parciales

Trabajo Práctico 4

Año 2020

Módulo 4

Problemas de Difusión

1. Condiciones de Borde

Asocie cada uno de los siguientes fragmentos de código a su correspondiente condición de borde:

(A)

```
u = numpy.ones(nx)
u[.5/dx : 1/dx+1] = 2

for n in range(1,nt):
    un = u.copy()
    u[1:-1] = un[1:-1] - c*dt/dx*(un[1:-1] - un[:-2])
```

(B)

```
u = numpy.ones(nx)
u[.5/dx : 1/dx+1] = 2

for n in range(1,nt):
    un = u.copy()
    u[1:-1] = un[1:-1] - c*dt/dx*(un[1:-1] - un[:-2])
    u[0] = u[1]
    u[-1] = u[-2]
```

(C)

```
u = numpy.ones(nx)
u[.5/dx : 1/dx+1] = 2

for n in range(1,nt):
    un = u.copy()
    u[1:-1] = un[1:-1] - c*dt/dx*(un[1:-1] - un[:-2])
    u[0] = un[0] - c*dt/dx*(un[0] - un[-1])
    u[-1] = un[-1] - c*dt/dx*(un[-1] - un[-2])
```

Condición de Borde

1. Neumann
2. Dirichlet
3. Robin
4. Periódica

2. Trabajo de código: Reacción-Difusión

Estudiaremos un modelo representado por ecuaciones de reacción-difusión. Consideramos en particular el modelo de *Gray-Scott*, el cual simula la interacción de dos especies químicas que reaccionan y difunden.

2.1. Modelo de Gray-Scott

Sean las especies U y V cuya concentración en un punto espacial se representa por las variables u y v. El modelo sigue reglas simples:

- cada especie química difunde a través del espacio a su propia velocidad;
- la especie U se alimenta al sistema a velocidad constante;
- dos unidades de especies V pueden transformar una unidad de especie U en V: $2V + U \to 3V$
- la especie V se remueve del sistema a velocidad constante.

Este modelo se describe por el siguiente sistema de PDEs para las concentraciones $u(x, y, t)$ y $v(x, y, t)$:

$$\frac{\partial u}{\partial t} = D_u \nabla^2 u - uv^2 + F(1 - u) \tag{1}$$

$$\frac{\partial v}{\partial t} = D_v \nabla^2 v + uv^2 - (F + k)v \tag{2}$$

El primer término del lado derecho de cada ecuación corresponde a la difusión espacial de cada concentración, siendo D_u y D_v las velocidades de difusión respectivas. Recordar el operador Laplaciano ∇^2:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{3}$$

El segundo término del lado derecho de cada ecuación corresponde a la reacción. Este término disminuye u e incrementa v en la misma cantidad: uv^2. La reacción requiere una unidad de U y dos unidades de V, resultando en una velocidad de reacción proporcional a la concentración u y al cuadrado de la concentración v.
El tercer término en ambas ecuaciones representa las velocidades de *alimentación* y *remoción*, respectivamente.

2.2. Datos del problema

El sistema se representa por dos vectores U y V los cuales almacenan los valores discretos de las concentraciones u y v, respectivamente. Con el propósito de construir la condición inicial, asignamos $U = 1$ y $V = 0$ en todo el dominio para luego introducir áreas de diferencia por medio de un pequeño ruido agregado al sistema completo. A modo de ejemplo, se ilustra el fragmento de código empleado:

```
num_blocks = 30
randx = numpy.random.randint(1, nx-1, num_blocks)
randy = numpy.random.randint(1, nx-1, num_blocks)
U = numpy.ones((n,n))
V = numpy.zeros((n,n))

r = 10
U[:,:] = 1.0

for i, j in zip(randx, randy):
    U[i-r:i+r,j-r:j+r] = 0.50
    V[i-r:i+r,j-r:j+r] = 0.25

U += 0.05*numpy.random.random((n,n))
V += 0.05*numpy.random.random((n,n))
```

Resolución de malla y Condiciones frontera:

- Discretize las ecuaciones de reacción-difusión empleando un esquema de adelanto en tiempo y centrado en espacio asumiendo $\Delta x = \Delta y = \delta$.
- Para el paso de tiempo, adopte

$$\Delta t = \frac{9}{40} \frac{\delta^2}{\text{máx}(D_u, D_v)} \tag{4}$$

- Adopte condiciones de Neumann homogéneas en todos los bordes del dominio.

El dominio se define por:

- Dimensión computacional (grilla de puntos): 192×192
- Dimensión física: 5m $\times$ 5m
- Instante final: 8000s

Utilize los siguientes parámetros:

```
import numpy
from matplotlib import pyplot, cm
%matplotlib inline

# Set spatial parameters
Lx, Ly = 5.0, 5.0  # domain dimensions
nx, ny = 192, 192  # number of points in each direction
dx, dy = Lx/(nx-1), Ly/(ny-1)  # grid spacings

# Set parameters of the pattern
Du, Dv = 0.00016, 0.00008  # rates of diffusion
F, k = 0.035, 0.065  # parameters to feed and kill

# Set temporal parameters
t = 8000.0  # final time
dt = 9.0*dx**2/(40.0*max(Du,Dv))  # time-step size
nt = int(t/dt)  # number of time steps to compute
```

2.3. Condición inicial

A fin de partir de las mismas condiciones iniciales (cada vez que se repita la simulación), procure leer el archivo uvinitial.npz el cual contiene los vectores U y V, empleando el siguiente fragmento de código:

```
# Read the initial fields from the file
filepath = 'uvinitial.npz'
uvinitial = numpy.load(filepath)
u0, v0 = uvinitial['U'], uvinitial['V']

# Plot the initial fields
fig, ax = pyplot.subplots(ncols=2, figsize=(9.0, 4.0))
ax[0].imshow(u0, cmap=cm.RdBu)
ax[0].axis('off')
ax[1].imshow(v0, cmap=cm.RdBu)
ax[1].axis('off');
```

2.4. Explorando patrones

Una vez que haya completado su simulación, explore algunos interesantes patrones que es posible obtener adoptando el conjunto de parámetros siguientes en el modelo de Gray-Scott:

```
#Du,Dv,F,k = 0.00014,0.00006,0.035,0.065 # Bacteria 2
#Du,Dv,F,k = 0.00016,0.00008,0.060,0.062 # Coral
#Du,Dv,F,k = 0.00019,0.00005,0.060,0.062 # Fingerprint
#Du,Dv,F,k = 0.00010,0.00010,0.018,0.050 # Spirals
#Du,Dv,F,k = 0.00012,0.00008,0.020,0.050 # Spirals Dense
#Du,Dv,F,k = 0.00010,0.00016,0.020,0.050 # Spirals Fast
#Du,Dv,F,k = 0.00016,0.00008,0.020,0.055 # Unstable
#Du,Dv,F,k = 0.00016,0.00008,0.050,0.065 # Worms 1
#Du,Dv,F,k = 0.00016,0.00008,0.054,0.063 # Worms 2
#Du,Dv,F,k = 0.00016,0.00008,0.035,0.060 # Zebrafish
```

2.5. Referencias

- Reaction-diffusion tutorial, by Karl Sims
- Pattern Parameters from aliensaint