Convección no lineal en dos dimensiones

Josué Juárez Morales

El siguiente paso es acoplar dos velocidades. Sean u y v el campo de velocidades en las direcciones x e y respectivamente, de modo que

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = 0, \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = 0, \tag{2}$$

la principal diferencia es que ahora tenemos dos ecuaciones diferenciales acopladas. Lo siguiente es similar, adimensionalizar y discretizar de forma separada para obtener

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta t} + v_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta t} = 0, \tag{3}$$

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta t} + v_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta t} = 0, \tag{4}$$

despejando para $u_{i,j}^{n+1}$, $v_{i,j}^{n+1}$ podemos avanzar en el tiempo

$$u_{i,j}^{n+1} = u_{i,j}^{n} - u_{i,j}\frac{\Delta t}{\Delta x}(u_{i,j}^{n} - u_{i-1,j}^{n}) - v_{i,j}^{n}\frac{\Delta t}{\Delta y}(u_{i,j}^{n} - u_{i,j-1}^{n}), \tag{5}$$

y

$$v_{i,j}^{n+1} = v_{i,j}^{n} - u_{i,j}\frac{\Delta t}{\Delta x}(v_{i,j}^{n} - v_{i-1,j}^{n}) - v_{i,j}^{n}\frac{\Delta t}{\Delta y}(v_{i,j}^{n} - v_{i,j-1}^{n}). \tag{6}$$

El siguiente ejemplo es para un pulso en dos dimensiones.

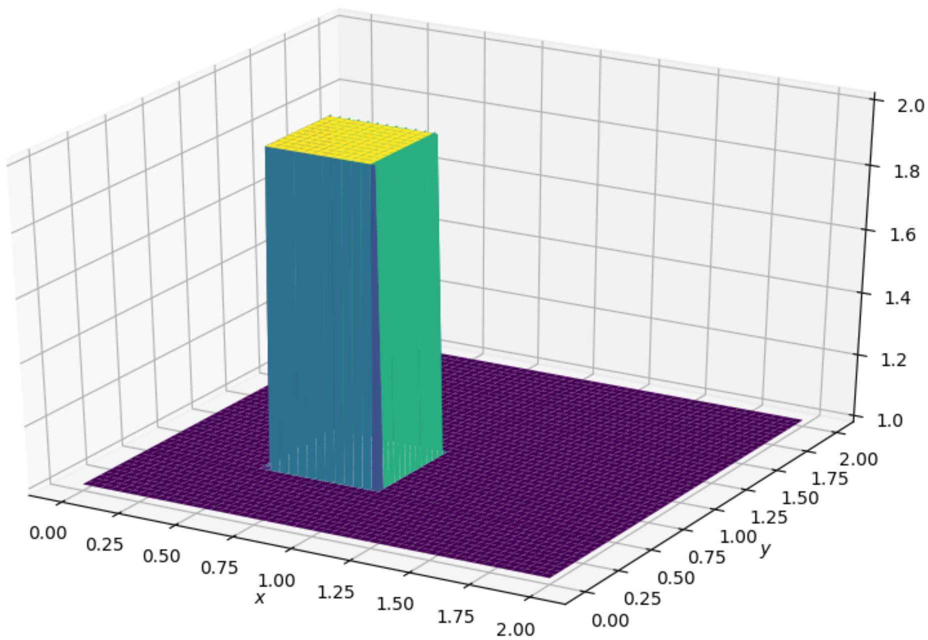
[0]:
```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

[0]:
```
def pulso(x0, x1, y0, y1, x, y):
  if (x0 < x and x < x1) and (y0 < y and y < y1):
    return 1.0
  else:
    return 0.0

#variables utiles
nx = 101
ny = 101
#nt = 80
```

```python
c = 1
Lx = 2
Ly = 2
dx = Lx / (nx-1)
dy = Ly / (ny -1)
CFL = .2
dt = CFL*dx

x = np.linspace(0, Lx, nx)
y = np.linspace(0, Ly, ny)

#vector de unos
u = np.ones((nx, ny))
v = np.ones((nx, ny))
un = np.ones((nx, ny))
vn = np.ones((nx, ny))

#condiciones iniciales
for i in range(nx):
  for j in range(ny):
    u[i,j] += pulso(0.5, 1.0, 0.5, 1.0, x[i], y[j])
    v[i,j] += pulso(0.5, 1.0, 0.5, 1.0, x[i], y[j])
```

[0]:
```python
fig = plt.figure(figsize=(11, 7), dpi = 100)
ax = fig.gca(projection='3d')
X, Y = np.meshgrid(x, y)

ax.plot_surface(X, Y, u, cmap=cm.viridis, rstride=2, cstride=2)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

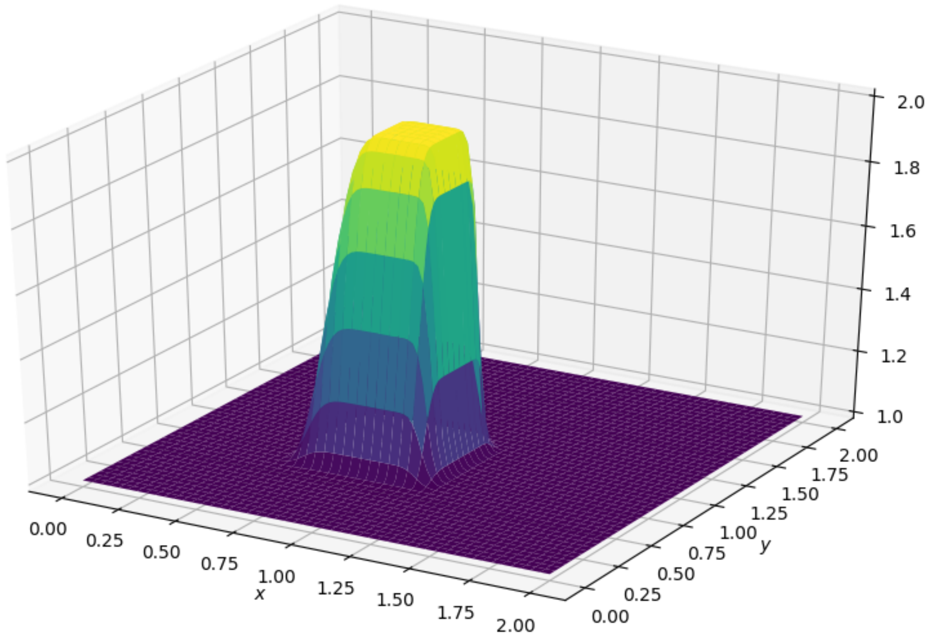
[0]:
```
Text(0.5, 0, '$y$')
```



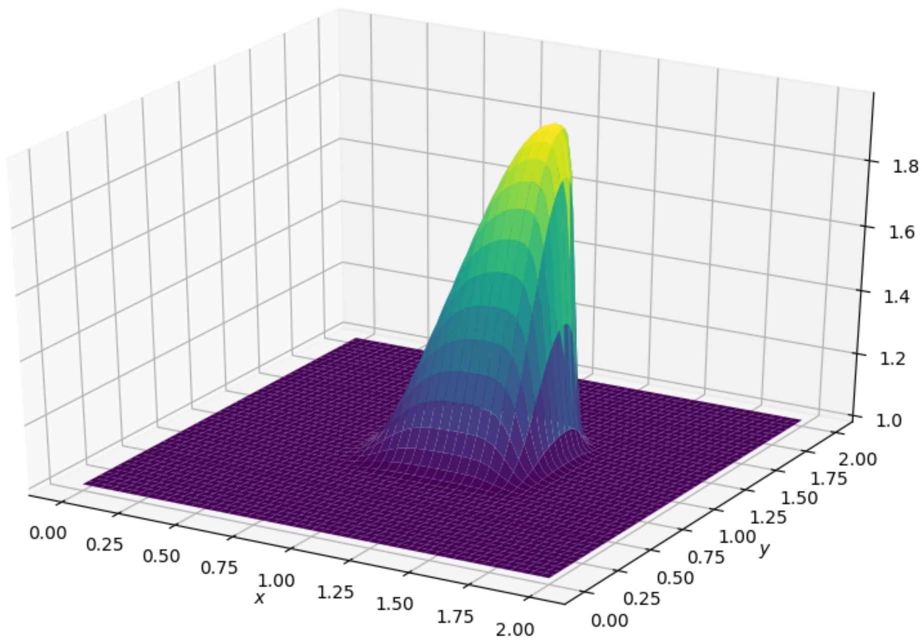
[0]:
```
def conveccion_no_lineal_2D(nt):
  for n in range(nt + 1):
    un = u.copy()
    vn = v.copy()
    for i in range(1, nx):
      for j in range(1, ny):
        u[i, j] = un[i, j] - un[i, j]*dt*(un[i, j] - un[i-1, j])/dx - vn[i,␣
↪j]*dt*(un[i, j] - un[i, j-1])/dy
        v[i, j] = vn[i, j] - vn[i, j]*dt*(un[i, j] - vn[i-1, j])/dx - vn[i,␣
↪j]*dt*(un[i, j] - vn[i, j-1])/dy

      #condiciones iniciales
      u[0,:] = 1.0
      u[-1,:] = 1.0
      u[:,0] = 1.0
      u[:,-1] = 1.0

      v[0,:] = 1.0
      v[-1,:] = 1.0
      v[:,0] = 1.0
      v[:,-1] = 1.0

  fig = plt.figure(figsize=(11, 7), dpi = 100)
  ax = fig.gca(projection='3d')
  X, Y = np.meshgrid(x, y)
```

```
    ax.plot_surface(X, Y, u, cmap=cm.viridis, rstride=2, cstride=2)
    ax.set_xlabel('$x$')
    ax.set_ylabel('$y$')
```

[0]:
```
conveccion_no_lineal_2D(10)
```



[0]:
```
conveccion_no_lineal_2D(50)
```



1.8
1.6
1.4
1.2
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
x
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
y