Ecuación de Burger en dos dimensiones

Josué Juárez Morales

La ecuación de Burger en dos dimensiones esta dada por el par de ecuaciones diferenciales parciales acopladas

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{2}$$

cada termino de estas ecuaciones las hemos discretizado en los pasos anteriores

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{u_{i,j}^{n} - u_{i,j-1}^{n}}{\Delta y} = \nu\left(\frac{u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\Delta x^2} + \frac{u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j+1}^{n}}{\Delta y^2}\right), \tag{3}$$

$$\frac{v_{i,j}^{n+1} - v_{i,j}^{n}}{\Delta t} + u_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i-1,j}^{n}}{\Delta x} + v_{i,j}^{n}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y} = \nu\left(\frac{v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}}{\Delta x^2} + \frac{v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j+1}^{n}}{\Delta y^2}\right), \tag{4}$$

lo unico que queda hace es despejar de estas ecuaciones los terminos $u_{i,j}^{n+1}$ y $v_{i,j}^{n+1}$ con los cuales podemos avanzar en el tiempo

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{\Delta t}{\Delta x}u_{i,j}^{n}(u_{i,j}^{n} - u_{i-1,j}^{n}) - \frac{\Delta t}{\Delta y}v_{i,j}^{n}(u_{i,j}^{n} - u_{i,j-1}^{n}) + \frac{\nu\Delta t}{\Delta x^2}(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}) + \frac{\nu\Delta t}{\Delta y^2}(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}), \tag{5}$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \frac{\Delta t}{\Delta x}u_{i,j}^{n}(v_{i,j}^{n} - v_{i-1,j}^{n}) - \frac{\Delta t}{\Delta y}v_{i,j}^{n}(v_{i,j}^{n} - v_{i,j-1}^{n}) + \frac{\nu\Delta t}{\Delta x^2}(v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}) + \frac{\nu\Delta t}{\Delta y^2}(v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}), \tag{6}$$

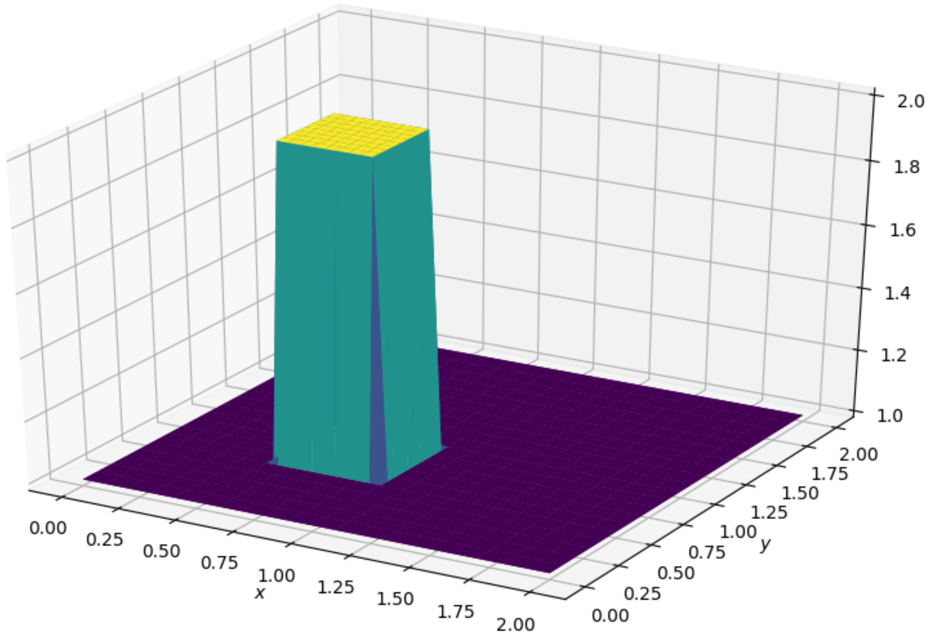
el siguiente es un ejemplo para la funcion pulso anterior.

[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

#declaración de variables
nx=41
ny=41
nt=120
nu= 0.05 #nu = 0.01 da error por overfloat
Lx =2.0
Ly =2.0
dx = Lx/ (nx-1)
dy= Ly/(nx-1)
CFL= 0.0009
dt=CFL*dx * dy/ nu

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])

fig = plt.figure(figsize=(11, 7), dpi = 100)
ax = fig.gca(projection='3d')
X, Y = np.meshgrid(x, y)
surf = ax.plot_surface(X, Y, u, rstride=1, cstride=1, cmap=cm.viridis)
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]:
```
for n in range(nt + 1):
  un = u.copy()
  vn = v.copy()

  for i in range(1, nx-1):
    for j in range(1, ny-1):
      u[i,j] = un[i,j] - dt*un[i,j]*(un[i,j] - un[i-1,j])/dx -␣
 ↪dt*vn[i,j]*(un[i,j] - un[i,j-1])/dy + nu*dt*(un[i+1,j] - 2.0*un[i,j] +␣
 ↪un[i-1,j])/(dx*dx) + nu*dt*(un[i,j+1] - 2.0*un[i,j] + un[i,j-1])/(dy*dy)
      v[i,j] = vn[i,j] - dt*un[i,j]*(vn[i,j] - vn[i-1,j])/dx -␣
 ↪dt*vn[i,j]*(vn[i,j] - vn[i,j-1])/dy + nu*dt*(vn[i+1,j] - 2.0*vn[i,j] +␣
 ↪vn[i-1,j])/(dx*dx) + nu*dt*(vn[i,j+1] - 2.0*vn[i,j] + vn[i,j-1])/(dy*dy)

u[0, :] = 1
u[-1, :] = 1
u[:, 0] = 1
u[:, -1] = 1

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

[0]:
```
fig = plt.figure(figsize=(11, 7), dpi = 100)
ax = fig.gca(projection='3d')
```

```
X, Y = np.meshgrid(x, y)
surf = ax.plot_surface(X, Y, u[:], rstride=1, cstride=1, cmap=cm.viridis)
#ax.plot_surface(X, Y, u, cmap=cm.viridis, rstride=2, cstride=2)
#ax.plot_surface(X, Y, v, cmap=cm.viridis, rstride=2, cstride=2)
ax.set_xlabel('$x$')
ax.set_ylabel('$y$')
```

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