CFD_2

August 27, 2024

diffusion equation in 2d:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} + \nu \frac{\partial^2 u}{\partial y^2}$$

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

Let us solve for $u_{i,j}^{n+1}$

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

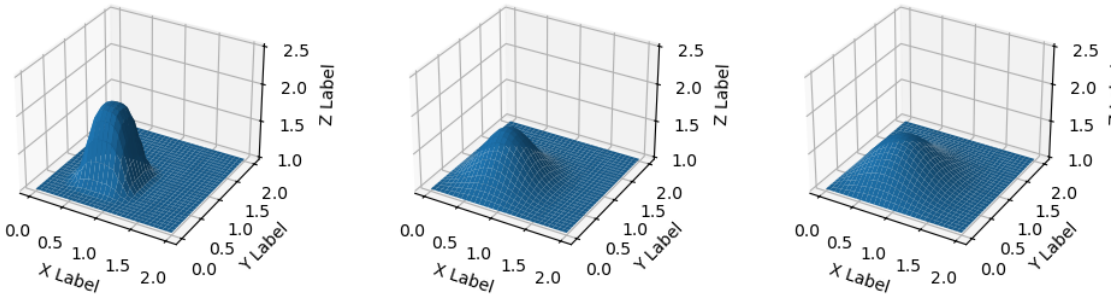
$$+ \frac{\nu \Delta t}{\Delta y^2}(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) \tag{2}$$

[1]:
```
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
from mpl_toolkits.mplot3d import axes3d, Axes3D
```

[2]:
```
###variable declarations
nx = 31
ny = 31
nt = 17
nu = .05
dx = 2 / (nx - 1)
dy = 2 / (ny - 1)
sigma = .25
dt = sigma * dx * dy / nu

x = np.linspace(0, 2, nx)
y = np.linspace(0, 2, ny)

u = np.ones((ny, nx))
un = np.ones((ny, nx))

u[int(.5 / dy):int(1 / dy + 1),int(.5 / dx):int(1 / dx + 1)] = 2
```

[3]:
```
def diffuse(nt):
    u[int(.5 / dy):int(1 / dy + 1),int(.5 / dx):int(1 / dx + 1)] = 2

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

[4]:
```
fig = plt.figure(figsize=(12,3))

ax = fig.add_subplot(1,3,1, projection='3d')
diffuse(3)
X, Y = np.meshgrid(x, y)
ax.set_zlim(1, 2.5)
ax.plot_surface(X, Y, u)
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

ax = fig.add_subplot(1,3,2, projection='3d')
diffuse(25)

X, Y = np.meshgrid(x, y)
ax.set_zlim(1, 2.5)
ax.plot_surface(X, Y, u)
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

ax = fig.add_subplot(1,3,3, projection='3d')
diffuse(40)

X, Y = np.meshgrid(x, y)
ax.set_zlim(1, 2.5)
ax.plot_surface(X, Y, u)
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

plt.show()
```



Laplace's pressure equation in 2D is written as

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = 0$$

The discretized form is written as

$$\frac{p^n_{i+1,j} - 2p^n_{i,j} + p^n_{i-1,j}}{\Delta x^2} + \frac{p^n_{i,j+1} - 2p^n_{i,j} + p^n_{i,j-1}}{\Delta y^2} = 0$$

The pressure equation is solved in each time step

$$p^n_{i,j} = \frac{\Delta y^2(p^n_{i+1,j} + p^n_{i-1,j}) + \Delta x^2(p^n_{i,j+1} + p^n_{i,j-1})}{2(\Delta x^2 + \Delta y^2)}$$

$\vec{v}$ represents the velocity field:

$$\nabla \cdot \vec{v} = 0 \tag{3}$$

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{v} \tag{4}$$

the continuity equation $\nabla \cdot \vec{v} = 0$ provides a that requires the pressure field to evolve so that the rate of expansion $\nabla \cdot \vec{v}$ should vanish everywhere. In incompressible flow construction of a pressure field guarantees continuity is satisfied.

$$\frac{p^n_{i+1,j} - 2p^n_{i,j} + p^n_{i-1,j}}{\Delta x^2} + \frac{p^n_{i,j+1} - 2p^n_{i,j} + p^n_{i,j-1}}{\Delta y^2} = b^n_{i,j}$$

As before, we rearrange this so that we obtain an equation for p at point i, j. Thus, we obtain:

$$p^n_{i,j} = \frac{(p^n_{i+1,j} + p^n_{i-1,j})\Delta y^2 + (p^n_{i,j+1} + p^n_{i,j-1})\Delta x^2 - b^n_{i,j}\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)}$$

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

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho\left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y}\right)$$

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

Similar for v

$$\frac{p_{i+1,j}^{n} - 2p_{i,j}^{n} + p_{i-1,j}^{n}}{\Delta x^2} + \frac{p_{i,j+1}^{n} - 2p_{i,j}^{n} + p_{i,j-1}^{n}}{\Delta y^2} =$$
$$\rho\left[\frac{1}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right) - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i}\right.$$

The momentum equation in the u direction:

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^{n} &- u_{i,j}^{n}\frac{\Delta t}{\Delta x}\left(u_{i,j}^{n} - u_{i-1,j}^{n}\right) - v_{i,j}^{n}\frac{\Delta t}{\Delta y}\left(u_{i,j}^{n} - u_{i,j-1}^{n}\right) \\ &- \frac{\Delta t}{\rho 2\Delta x}\left(p_{i+1,j}^{n} - p_{i-1,j}^{n}\right) \\ &+ \nu\left(\frac{\Delta t}{\Delta x^2}\left(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + \frac{\Delta t}{\Delta y^2}\left(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}\right)\right) \end{aligned}$$

The momentum equation in the v direction:

$$\begin{aligned} v_{i,j}^{n+1} = v_{i,j}^{n} &- u_{i,j}^{n}\frac{\Delta t}{\Delta x}\left(v_{i,j}^{n} - v_{i-1,j}^{n}\right) - v_{i,j}^{n}\frac{\Delta t}{\Delta y}\left(v_{i,j}^{n} - v_{i,j-1}^{n}\right)) \\ &- \frac{\Delta t}{\rho 2\Delta y}\left(p_{i,j+1}^{n} - p_{i,j-1}^{n}\right) \\ &+ \nu\left(\frac{\Delta t}{\Delta x^2}\left(v_{i+1,j}^{n} - 2v_{i,j}^{n} + v_{i-1,j}^{n}\right) + \frac{\Delta t}{\Delta y^2}\left(v_{i,j+1}^{n} - 2v_{i,j}^{n} + v_{i,j-1}^{n}\right)\right) \end{aligned}$$

Rearranging the pressure-Poisson equation:

$$\begin{aligned} p_{i,j}^{n} = &\frac{\left(p_{i+1,j}^{n} + p_{i-1,j}^{n}\right)\Delta y^2 + \left(p_{i,j+1}^{n} + p_{i,j-1}^{n}\right)\Delta x^2}{2\left(\Delta x^2 + \Delta y^2\right)} \\ &- \frac{\rho\Delta x^2\Delta y^2}{2\left(\Delta x^2 + \Delta y^2\right)} \\ &\times\left[\frac{1}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right) - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - 2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \right. \end{aligned}$$

[5]:
```
nx = 41
ny = 41
nt = 500
nit = 50
c = 1
dx = 2 / (nx - 1)
dy = 2 / (ny - 1)
x = np.linspace(0, 2, nx)
y = np.linspace(0, 2, ny)
X, Y = np.meshgrid(x, y)

rho = 1
nu = .1
dt = .001

u = np.zeros((ny, nx))
v = np.zeros((ny, nx))
p = np.zeros((ny, nx))
b = np.zeros((ny, nx))
```

[6]:
```
def build_up_b(b, rho, dt, u, v, dx, dy):

    b[1:-1, 1:-1] = (rho * (1 / dt *
                    ((u[1:-1, 2:] - u[1:-1, 0:-2]) /
                     (2 * dx) + (v[2:, 1:-1] - v[0:-2, 1:-1]) / (2 * dy)) -
                    ((u[1:-1, 2:] - u[1:-1, 0:-2]) / (2 * dx))**2 -
                      2 * ((u[2:, 1:-1] - u[0:-2, 1:-1]) / (2 * dy) *
                           (v[1:-1, 2:] - v[1:-1, 0:-2]) / (2 * dx))-
                          ((v[2:, 1:-1] - v[0:-2, 1:-1]) / (2 * dy))**2))

    return b
```

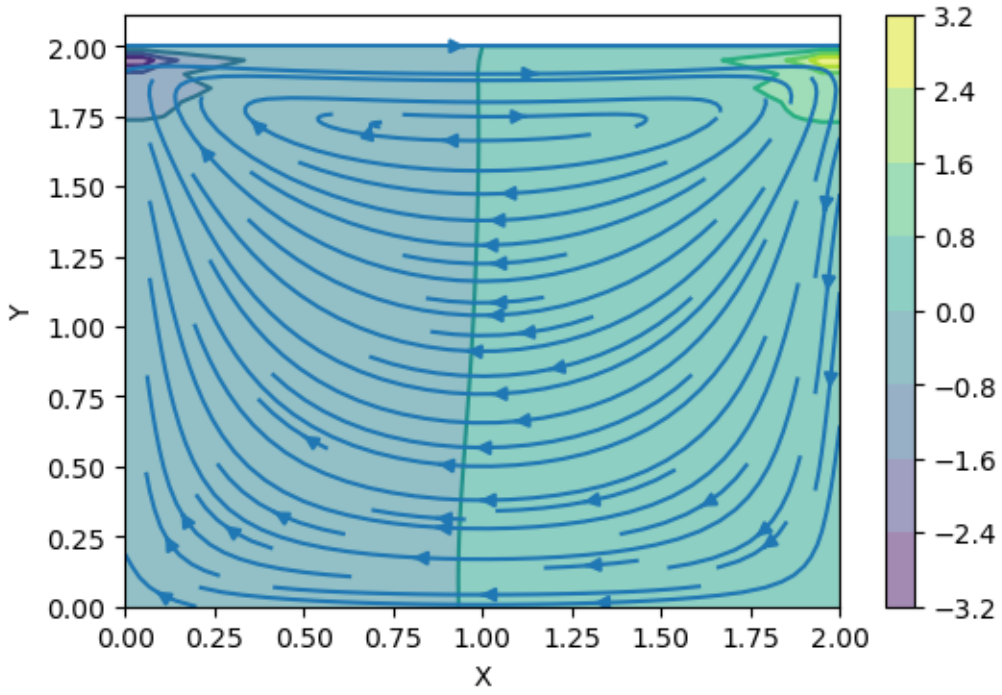
[7]:
```
def pressure_poisson(p, dx, dy, b):
    pn = np.empty_like(p)
    pn = p.copy()

    for q in range(nit):
        pn = p.copy()
        p[1:-1, 1:-1] = (((pn[1:-1, 2:] + pn[1:-1, 0:-2]) * dy**2 +
                          (pn[2:, 1:-1] + pn[0:-2, 1:-1]) * dx**2) /
                          (2 * (dx**2 + dy**2)) -
                          dx**2 * dy**2 / (2 * (dx**2 + dy**2)) *
                          b[1:-1,1:-1])

        p[:, -1] = p[:, -2] # dp/dx = 0 at x = 2
        p[0, :] = p[1, :]   # dp/dy = 0 at y = 0
        p[:, 0] = p[:, 1]   # dp/dx = 0 at x = 0
```

```
        p[-1, :] = 0        # p = 0 at y = 2

    return p
```

[8]:
```
def cavity_flow(nt, u, v, dt, dx, dy, p, rho, nu):
    un = np.empty_like(u)
    vn = np.empty_like(v)
    b = np.zeros((ny, nx))

    for n in range(nt):
        un = u.copy()
        vn = v.copy()

        b = build_up_b(b, rho, dt, u, v, dx, dy)
        p = pressure_poisson(p, dx, dy, b)

        u[1:-1, 1:-1] = (un[1:-1, 1:-1]-
                         un[1:-1, 1:-1] * dt / dx *
                        (un[1:-1, 1:-1] - un[1:-1, 0:-2]) -
                         vn[1:-1, 1:-1] * dt / dy *
                        (un[1:-1, 1:-1] - un[0:-2, 1:-1]) -
                         dt / (2 * rho * dx) * (p[1:-1, 2:] - p[1:-1, 0:-2]) +
                         nu * (dt / dx**2 *
                        (un[1:-1, 2:] - 2 * un[1:-1, 1:-1] + un[1:-1, 0:-2]) +
                         dt / dy**2 *
                        (un[2:, 1:-1] - 2 * un[1:-1, 1:-1] + un[0:-2, 1:-1])))

        v[1:-1,1:-1] = (vn[1:-1, 1:-1] -
                        un[1:-1, 1:-1] * dt / dx *
                       (vn[1:-1, 1:-1] - vn[1:-1, 0:-2]) -
                        vn[1:-1, 1:-1] * dt / dy *
                       (vn[1:-1, 1:-1] - vn[0:-2, 1:-1]) -
                        dt / (2 * rho * dy) * (p[2:, 1:-1] - p[0:-2, 1:-1]) +
                        nu * (dt / dx**2 *
                       (vn[1:-1, 2:] - 2 * vn[1:-1, 1:-1] + vn[1:-1, 0:-2]) +
                        dt / dy**2 *
                       (vn[2:, 1:-1] - 2 * vn[1:-1, 1:-1] + vn[0:-2, 1:-1])))

        u[0, :]  = 0
        u[:, 0]  = 0
        u[:, -1] = 0
        u[-1, :] = 1    # set velocity on cavity lid equal to 1
        v[0, :]  = 0
        v[-1, :] = 0
        v[:, 0]  = 0
        v[:, -1] = 0
```

```
    return u, v, p
```

[9]:
```
u = np.zeros((ny, nx))
v = np.zeros((ny, nx))
p = np.zeros((ny, nx))
b = np.zeros((ny, nx))
nt = 100
u, v, p = cavity_flow(nt, u, v, dt, dx, dy, p, rho, nu)

fig = plt.figure(figsize=(6,4))
plt.contourf(X, Y, p, alpha=0.5, cmap=cm.viridis)
plt.colorbar()
plt.contour(X, Y, p, cmap=cm.viridis)
plt.streamplot(X, Y, u, v)
plt.xlabel('X')
plt.ylabel('Y');
```



[ ]: