Climate Modeling

November 11, 2024

PRACTICAL REPORT

DIFFUSION: BONUS

By:

Duong Thu Phuong

22BI13362

Simulate heat diffusion on a 2D plate using Crank-Nicholson scheme. There are 2 heaters at the initial conditions. Boundary conditions are zero at every timestep.

At the initial time, t=0:

T[30:50,30:50] = 1

T[60:80,60:80] = 0.8

T=0 elsewhere

[1]:
```
import numpy as np
from scipy.sparse import diags, csr_matrix
from scipy.sparse.linalg import spsolve
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation, PillowWriter
from IPython.display import Image
```

The initial is the same as previous practice, except that I incease lenX and lenY to 100.

[2]:
```
lenX = 100
lenY = 100
k = 0.05
dt = 0.1
dx = 0.5
dy = 0.5

Nx = int(lenX / dx) + 1
Ny = int(lenY / dy) + 1
```

[3]:
```
def initial(nx, ny):
    T = np.zeros((ny, nx))
    T[30:50, 30:50] = 1.0
    T[60:80, 60:80] = 0.8
    return T
```

1D advection-diffusion equation:

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} + u\frac{\partial C}{\partial x} + bC$$

Generalized formulation:

$$\frac{C_{m,n+1} - C_{m,n}}{\Delta t} = D\left(\theta\frac{\nabla_x^2 C_{m,n+1}}{\Delta x^2} + (1-\theta)\frac{\nabla_x^2 C_{m,n}}{\Delta x^2}\right) + u\frac{\nabla_x C_{m,n}}{2\Delta x} + bC_{m,n}$$

with two central difference operators:

$$\nabla_x C_{m,n} = C_{m+1,n} - C_{m-1,n}$$

$$\nabla_x^2 C_{m,n} = C_{m+1,n} - 2C_{m,n} + C_{m-1,n}$$

When $u = 0$ and $b = 0$, the Crank-Nicholson Scheme simplifies to:

$$\frac{C_{m,n+1} - C_{m,n}}{\Delta t} = D\left(\frac{1}{2}\frac{\nabla_x^2 C_{m,n+1}}{\Delta x^2} + \frac{1}{2}\frac{\nabla_x^2 C_{m,n}}{\Delta x^2}\right)$$

I'm too lazy to continue typing these formulas (it really hurts my eyes and my fingers) so Imma just scan my handwriting here.

$$\frac{C_{m,n+1} - C_{m,n}}{\Delta t} = D\left(\frac{1}{2}\frac{\nabla_x^2 C_{m,n+1}}{\Delta x^2} + \frac{1}{2}\frac{\nabla_x^2 C_{m,n}}{\Delta x^2}\right) \to 1D$$

2D heat equation: $\dfrac{\partial C}{\partial t} = D\left(\dfrac{\partial^2 C}{\partial x^2} + \dfrac{\partial^2 C}{\partial y^2}\right)$

call $C^n_{i,j}$ as concentration at time n @ grid point i,j

$$C^{n+1}_{i,j} - C^n_{i,j} = \Delta t \left.\frac{\partial C}{\partial t}\right|_{i,j,n}$$

central difference: $\dfrac{\partial^2 C}{\partial x^2} = \dfrac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta x^2}$

$$\frac{\partial^2 C}{\partial y^2} = \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta y^2}$$

$$\to \frac{C^{n+1}_{i,j} - C^n_{i,j}}{\Delta t} = D\left(\frac{C_{i+1,j} - 2C_{i,j} + C_{i-1,j}}{\Delta x^2} + \frac{C_{i,j+1} - 2C_{i,j} + C_{i,j-1}}{\Delta y^2}\right)$$

Crank Nicholson:

$$\frac{C^{n+1}_{i,j} - C^n_{i,j}}{\Delta t} = D\left(\frac{1}{2}\left(\frac{C^{n+1}_{i+1,j} - 2C^{n+1}_{i,j} + C^{n+1}_{i-1,j}}{\Delta x^2} + \frac{C^{n+1}_{i,j+1} - 2C^{n+1}_{i,j} + C^{n+1}_{i,j-1}}{\Delta y^2}\right)\right)$$

$$+ D\left(\frac{1}{2}\left(\frac{C^n_{i+1,j} - 2C^n_{i,j} + C^n_{i-1,j}}{\Delta x^2} + \frac{C^n_{i,j+1} - 2C^n_{i,j} + C^n_{i,j-1}}{\Delta y^2}\right)\right)$$

$$\to C^{n+1}_{i,j} = C^n_{i,j} + \frac{D\Delta t}{2\Delta x^2}\left(C^{n+1}_{i+1,j} - 2C^{n+1}_{i,j} + C^{n+1}_{i-1,j} + C^n_{i+1,j} - 2C^n_{i,j} + C^n_{i-1,j}\right)$$

$$+ \frac{D\Delta t}{2\Delta y^2}\left(C^{n+1}_{i,j+1} - 2C^{n+1}_{i,j} + C^{n+1}_{i,j-1} + C^n_{i,j+1} - 2C^n_{i,j} + C^n_{i,j-1}\right)$$

Set $\dfrac{D\Delta t}{2\Delta x^2} = r_x$ and $\dfrac{D\Delta t}{2\Delta y^2} = r_y$

$$\to C^{n+1}_{i,j} - r_x\left(C^{n+1}_{i+1,j} + C^{n+1}_{i-1,j}\right) - r_y\left(C^{n+1}_{i,j+1} + C^{n+1}_{i,j-1}\right) = C^n_{i,j} + r_x\left(C^n_{i+1,j}\right.$$

$$\left. + C^n_{i-1,j}\right) + r_y\left(C^n_{i,j+1} + C^n_{i,j-1}\right)$$

$$C^n = \begin{bmatrix} C^n_{1,1} \\ C^n_{1,2} \\ \vdots \\ C^m_{M,N} \end{bmatrix} \qquad C^{n+1} = \begin{bmatrix} C^{n+1}_{1,1} \\ C^{n+1}_{1,2} \\ \vdots \\ C^{n+1}_{M,N} \end{bmatrix}$$

$$A\overrightarrow{C^{n+1}} = B\overrightarrow{C^{n}}$$

$$A = \begin{bmatrix} 1+2r_x+2r_y & -r_x & 0 & \cdots & 0 & -r_y & 0 \\ & & & & & 0 & -r_y \\ -r_x & 1+2r_x+2r_y & -r_x & \cdots & 0 & 0 & -r_y \\ 0 & -r_x & 1+2r_x+2r_y & \cdots & 0 & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1+2r_x+2r_y & -r_x & -r_y \\ -r_y & 0 & 0 & \cdots & -r_x & 1+2r_x+2r_y & -r_y \\ 0 & -r_y & 0 & \cdots & 0 & -r_x & 1+2r_x+2r_y \end{bmatrix}$$

$$B = \begin{bmatrix} C^n_{1,1} + r_x(C^n_{0,1} + C^n_{2,1}) + r_y(C^n_{1,0} + C^n_{1,2}) \\ C^n_{1,2} + r_x(C^n_{0,2} + C^n_{2,2}) + r_y(C^n_{1,1} + C^n_{1,3}) \\ \vdots \\ C^m_{M,N} + r_x(C^n_{M-1,N} + C^n_{M+1,N}) + r_y(C^n_{M,N-1} + C^n_{M,N+1}) \end{bmatrix}$$

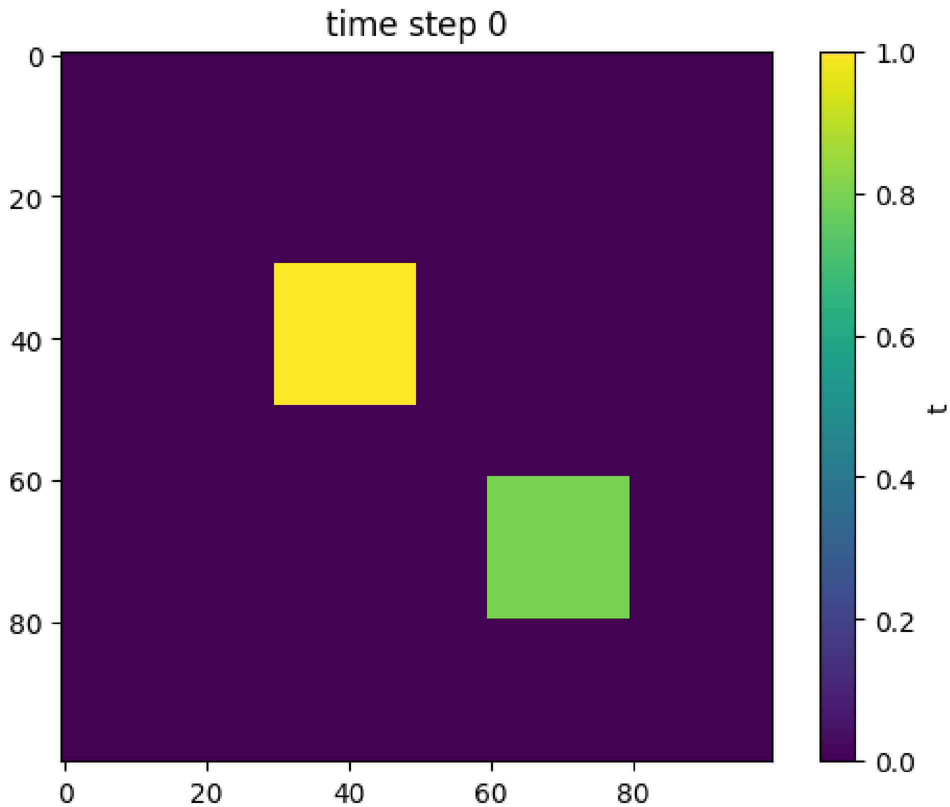
[4]:
```python
steps = 1000

def cranknicolson(nx, ny, rx, ry):
    N = nx * ny

    # Matrix A: (1 + 2rx + 2ry)I - rx(Ex + Ew) - ry(En + Es)
    main = np.ones(N) * (1 + 2*rx + 2*ry)
    diag_x = np.ones(N-1) * (-rx)
    diag_y = np.ones(N-ny) * (-ry)

    # Remove connections between rows
    for i in range(ny-1):
        diag_x[(i+1)*nx-1] = 0

    diag_A = [main, diag_x, diag_x, diag_y, diag_y]
    offsets = [0, 1, -1, ny, -ny]
    A = diags(diag_A, offsets, format='csr')

    # Matrix B: I + rx(Ex + Ew) + ry(En + Es)
```

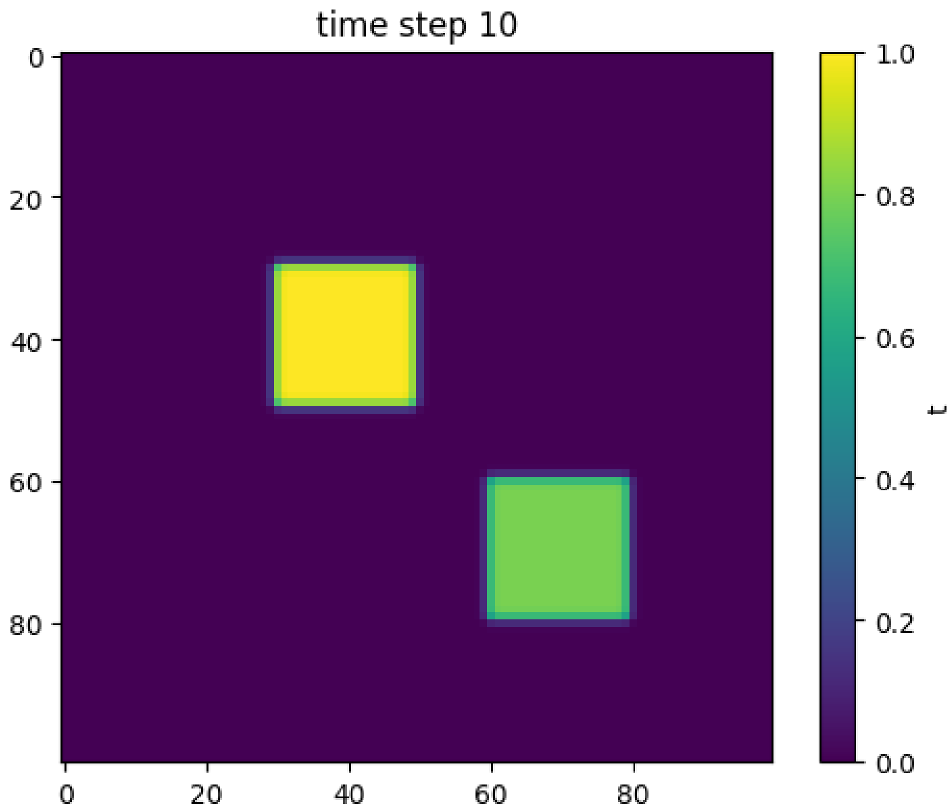
```
    main_B = np.ones(N) * (1 - 2*rx - 2*ry)
    diag_x_B = np.ones(N-1) * rx
    diag_y_B = np.ones(N-ny) * ry

    for i in range(ny-1):
        diag_x_B[(i+1)*nx-1] = 0

    diag_B = [main_B, diag_x_B, diag_x_B, diag_y_B, diag_y_B]
    B = diags(diag_B, offsets, format='csr')

    return A, B
```

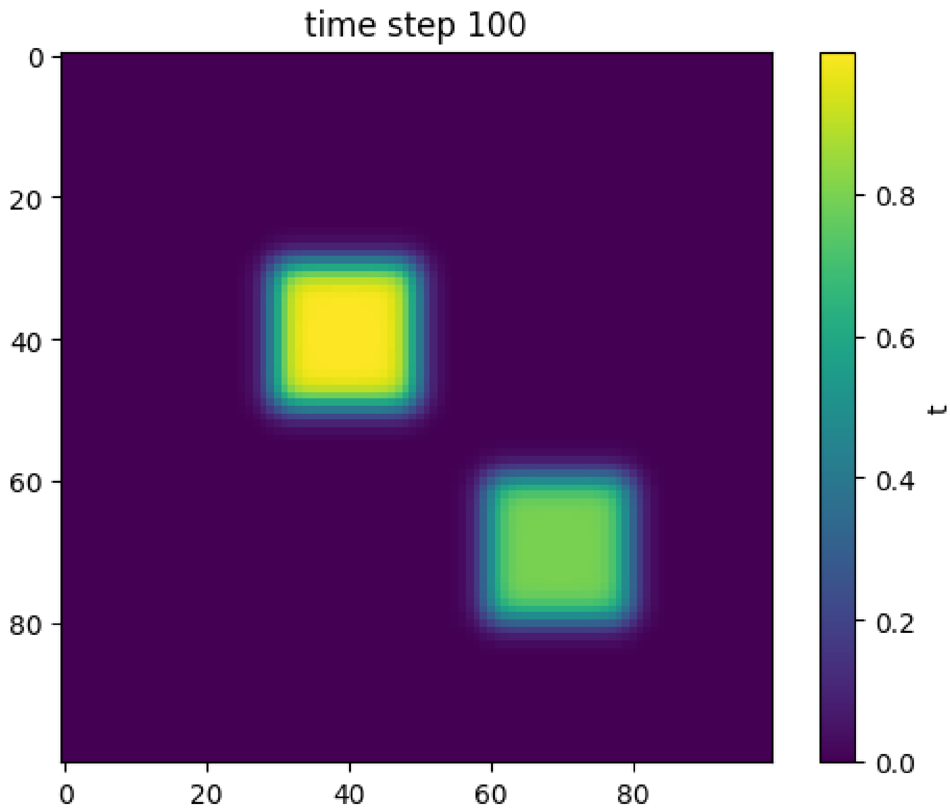
[5]:
```
def heat(nx, ny, k, dt, dx, dy, max_steps):
    rx = k * dt / (2 * dx**2)
    ry = k * dt / (2 * dy**2)
    T = initial(nx, ny)
    A, B = cranknicolson(nx, ny, rx, ry)
    results = [T.copy()]

    for step in range(steps):
        T_flat = T.reshape(-1)

        T_flat[0:nx] = 0
        T_flat[-nx:] = 0
        T_flat[::nx] = 0
        T_flat[nx-1::nx] = 0

        # AC(n+1) = BCn
        b = B.dot(T_flat)
        T_new = spsolve(A, b)

        # Reshape back to 2D
        T = T_new.reshape((ny, nx))
        T[0, :] = T[-1, :] = T[:, 0] = T[:, -1] = 0

        if step in [9, 99, 999, 9999]:
            results.append(T.copy())

    return results

results = heat(lenX, lenY, k, dt, dx, dy, steps)
```

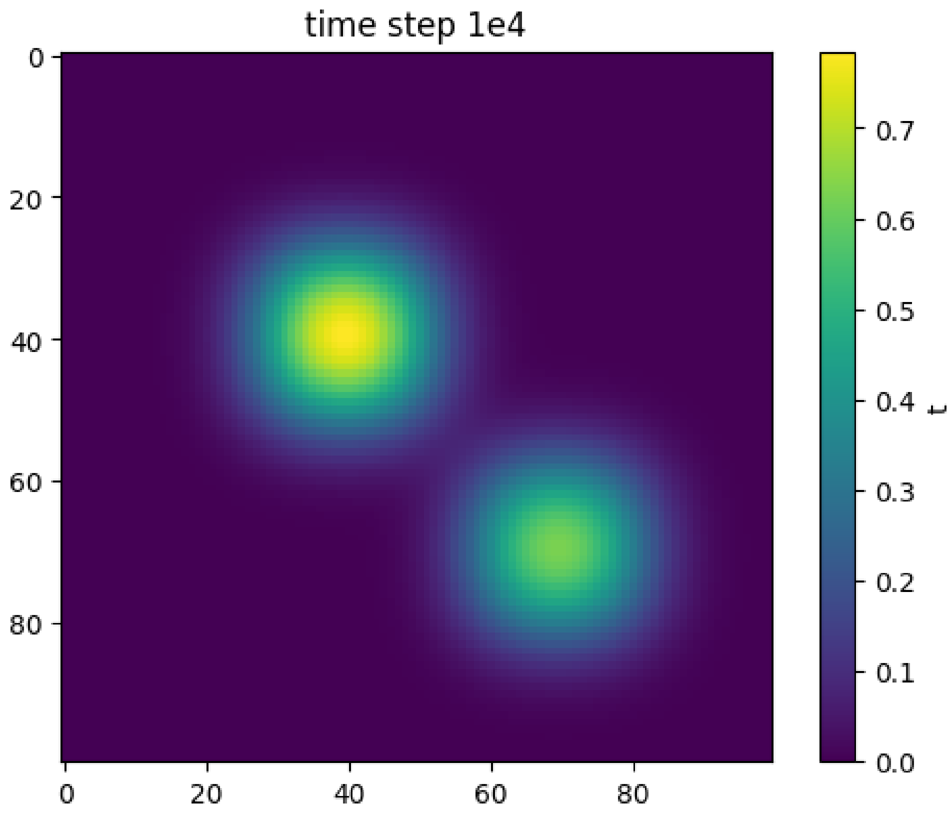
[6]:
```
times = [0, 1, 2, 3]
titles = ['time step 0',
          'time step 10',
          'time step 100',
          'time step 1e4']
```

```
for T, title in zip(results, titles):
    plt.figure()
    im = plt.imshow(T)
    plt.colorbar(label='t')
    plt.title(title)
    plt.show()
```




time step 10
0
20
40
60
80
0
20
40
60
80
1.0
0.8
0.6
0.4
0.2
0.0
t


time step 100
0
20
40
60
80
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
t



[7]:
```
for T, title in zip(results, titles):
    time = titles.index(title) * 50 * dt
    print(f"\n{title}:")
    print(f"maximum temperature: {np.max(T)}")
    print(f"average temperature: {np.mean(T)}")
```

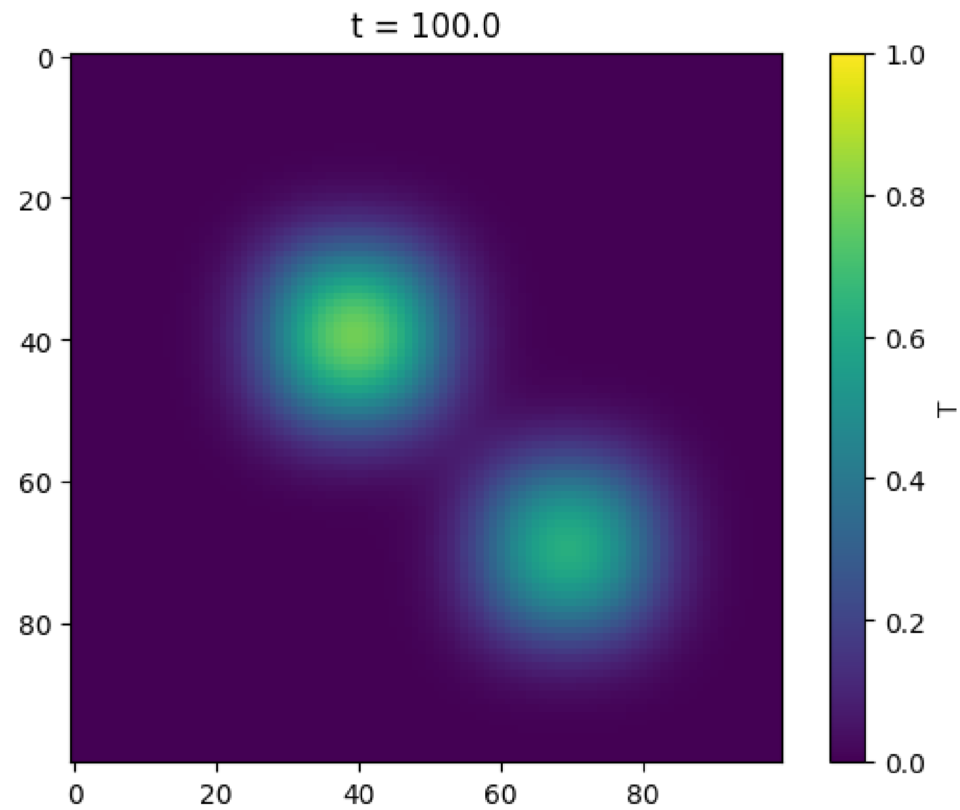
```
time step 0:
maximum temperature: 1.0
average temperature: 0.072

time step 10:
maximum temperature: 0.9999999999999375
average temperature: 0.07199999999999993

time step 100:
maximum temperature: 0.9999789568027068
average temperature: 0.07199999999999915

time step 1e4:
maximum temperature: 0.7842612965925905
```

```
average temperature: 0.07198715680527387
```

I plot the maximum temperature and average temperature for initial stage, the stage for 10 and 100 and 1000 time steps. As being seen, the average temperature remains quite stable. The maximum temperature decreases as the progress going, because heat moves from area of higher concentration to lower, and the energy spreads out.

[9]:
```python
def heat(nx, ny, k, dt, dx, dy, steps):
    rx = k * dt / (2 * dx**2)
    ry = k * dt / (2 * dy**2)
    T = initial(nx, ny)
    A, B = cranknicolson(nx, ny, rx, ry)
    frames = [(T.copy(), 0)]

    for step in range(1, steps + 1):
        T_flat = T.reshape(-1)

        T_flat[0:nx] = 0
        T_flat[-nx:] = 0
        T_flat[::nx] = 0
        T_flat[nx-1::nx] = 0

        b = B.dot(T_flat)
        T_new = spsolve(A, b)
        T = T_new.reshape((ny, nx))
        T[0, :] = T[-1, :] = T[:, 0] = T[:, -1] = 0

        frames.append((T.copy(), step * dt))

    return frames

results = heat(lenX, lenY, k, dt, dx, dy, steps)

fig, ax = plt.subplots()
im = ax.imshow(results[0][0])
title = ax.set_title(f't = {results[0][1]}')
plt.colorbar(im, ax=ax, label='T')

def update(frame):
    T, time = frame
    im.set_array(T)
    title.set_text(f't = {time:.1f}')
    return [im, title]

ani = FuncAnimation(fig, update, frames=results, blit=True)
ani.save('heat_diffusion.gif', writer=PillowWriter(fps=100))

plt.show()
```

```python
Image(filename='heat_diffusion.gif')
```



[9]: <IPython.core.display.Image object>

As being seen, as the heat diffusion progresses, the boundaries of the heater spread out. At the initial stages, the heaters are highly localized and have sharp contrasts between them and the surrounding area. Later frames depict smoother, more distributed heat over all the area, indicating that the system is balancing out.

I cannot submit the .gif file on Moodle so I upload it here instead.

[ ]:

```python

```