Project 4 part 2

Georgios (George) Sevastakis

November 2023

Introducing the code

The code is structured in the following way: Two functions are defined, namely the *Laplacian* and *Euler* functions. The former computes the Laplacian of a given matrix by implementing the finite difference method, while the latter boosts the p and q (mathematical) functions in time using the forward Euler method. The Neumann boundary conditions are to be found in the *Euler* function

Laplacian

```
def Laplacian(M): #Function which computes the Laplacian of a given matrix M

    CM = np.copy(M)
    CM = (np.roll(CM, -1, axis=0) +np.roll(CM, 1, axis=0)-2*CM)/(dx**2) + (np.
   roll(CM, -1, axis=1) +np.roll(CM, 1, axis=1)-2*CM)/(dy**2)

    return CM
```

Laplacian function

As stated above, finite difference method is utilized in order to approximate the Laplacian. In particular, we approximated the Laplacian at the point (x, y) as follows

$$\nabla^2 M(x,y) = \frac{\partial^2 M(x,y)}{\partial x^2} + \frac{\partial^2 M(x,y)}{\partial y^2}$$

$$\Rightarrow \nabla^2 M \approx \frac{M(x-dx,y) - 2M(x,y) + M(x+dx,y)}{dx^2} + \frac{M(x,y-dy) - 2M(x,y) + M(x,y+dy)}{dy^2}$$

The above approximation is a second-order central difference approximation.

We managed to avoid the $for-$loops in our code by implementing the command *roll*, which shifts a particular axis by the step of our choice. Therefore, we were able to calculate the Laplacian of each point in one go. Boundary conditions will be discussed later.

Euler

```
def Euler(N, K):
        C=4.5

        p=np.zeros((N,N))
        q=np.zeros((N,N))

        for l in range (N):
            for ll in range (N):
                if x[l]>10 and x[l]<30 and y[ll]>10 and y[ll]<30:  #initial
    conditions
                    p[l,ll]=C+0.1
                    q[l,ll]=(K/C)+0.2

        dt=0.01  #time step
        T=np.zeros(2000)

        Dp=1. #diffusion coefficient of p
        Dq=8. #diffusion coefficient of q

        for i in range (len(T)):

            p+=(Dp*Laplacian(p)+(p*p)*q+C-(K+1)*p)*dt  #Euler algorithm for p
            q+=(Dq*Laplacian(q)-(p*p)*q+K*p)*dt     #Euler algorithm for q

            p[0,:]=p[2,:]           #Neumann boundary conditions
            p[N-1,:]= p[N-3,:]
            p[:,0]= p[:,2]
            p[:,N-1]= p[:,N-3]

            q[0,:]= q[2,:]
            q[N-1,:]= q[N-3,:]
            q[:,0]=  q[:,2]
            q[:,N-1]=q[:,N-3]

        return p, q
```

Euler function

By calling the *Euler* function, not only do we boost p and q in time, but we are also applying the Neumann boundary conditions. To be more specific, *Euler* takes the number of grid points of one axis and the constant K as arguments and returns the solutions p and q with the help of the *Laplacian* function after $t = 2000$.

As far as the Neumann boundary conditions are concerned, we want the net flux at the borders to be zero, meaning that the directional derivatives at the boundary ought to be zero. In our case, the boundary is a square and therefore, we demand for each of the directional derivatives of p and q to be zero at the borders of the square, meaning

$$\left.\frac{\partial p}{\partial x}\right|_{x=0} = \left.\frac{\partial q}{\partial x}\right|_{x=0} = \left.\frac{\partial p}{\partial x}\right|_{x=x_N} = \left.\frac{\partial q}{\partial x}\right|_{x=x_N} = 0$$

and

$$\left.\frac{\partial p}{\partial y}\right|_{y=0} = \left.\frac{\partial q}{\partial y}\right|_{y=0} = \left.\frac{\partial p}{\partial y}\right|_{y=y_N} = \left.\frac{\partial q}{\partial y}\right|_{y=y_N} = 0$$

Translated into code, we implemented the first order central difference method for each one of them

$$\left.\frac{\partial M(x,y)}{\partial x}\right|_{\partial} \approx \frac{M(x+dx,y)\big|_{\partial} - M(x-dx,y)\big|_{\partial}}{2dx} = 0$$

$$\Rightarrow \left. M(x+dx,y)\right|_{\partial} = \left. M(x-dx,y)\right|_{\partial}$$

Equivalently,

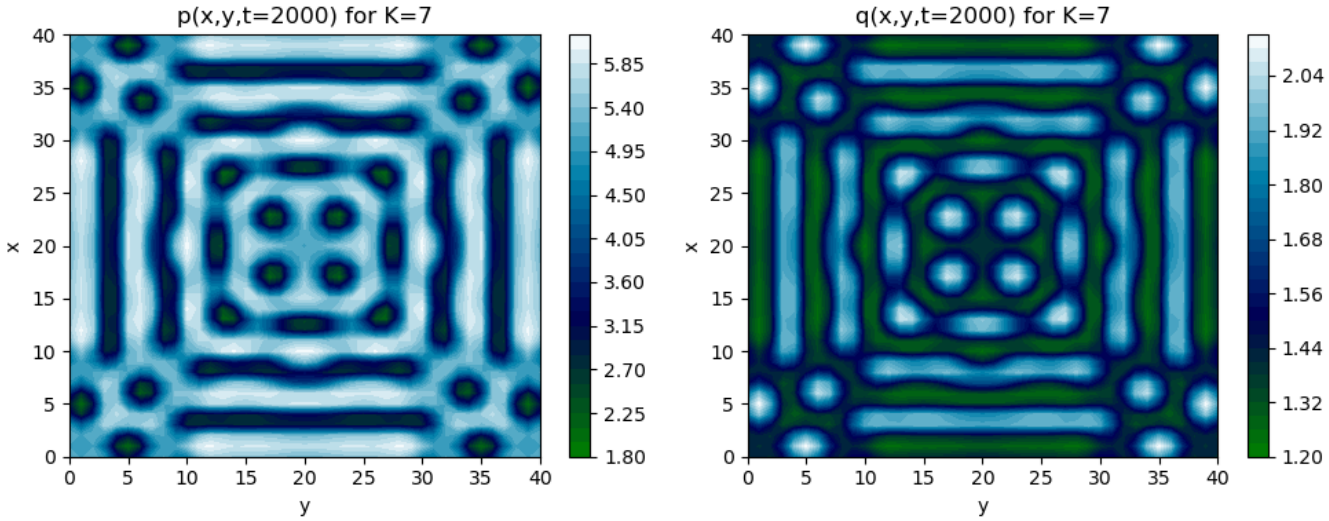
$$\left. M(x,y+dy)\right|_{\partial} = \left. M(x,y-dy)\right|_{\partial}$$

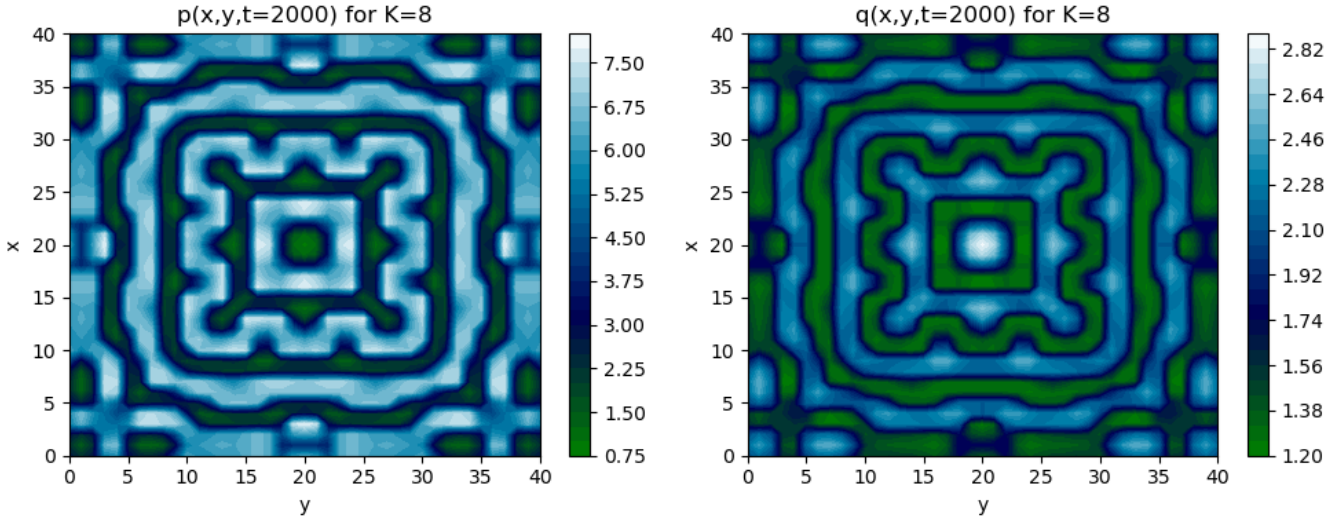
where $\big|_{\partial}$ denotes the boundary.

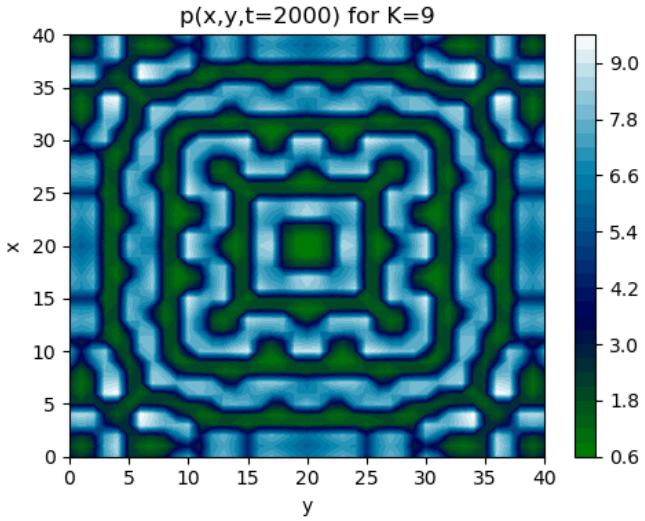
Results

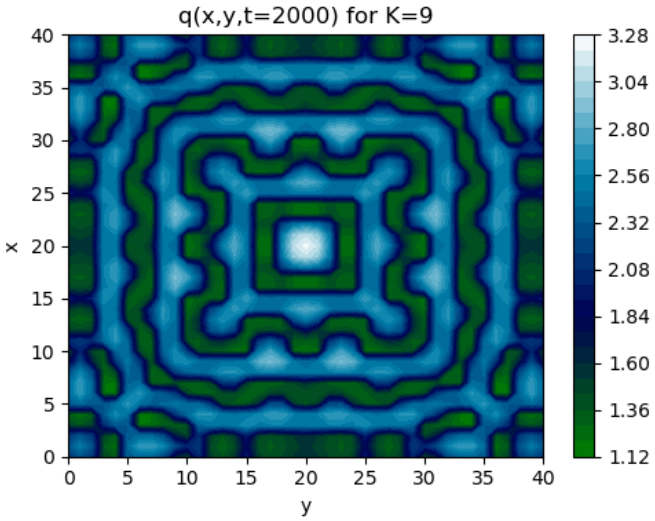
Finally, we demonstrate the contours of p and q for different values of the parameters D_p, D_q, C, K, dt and N (number of grid points of a single line of the mesh grid)

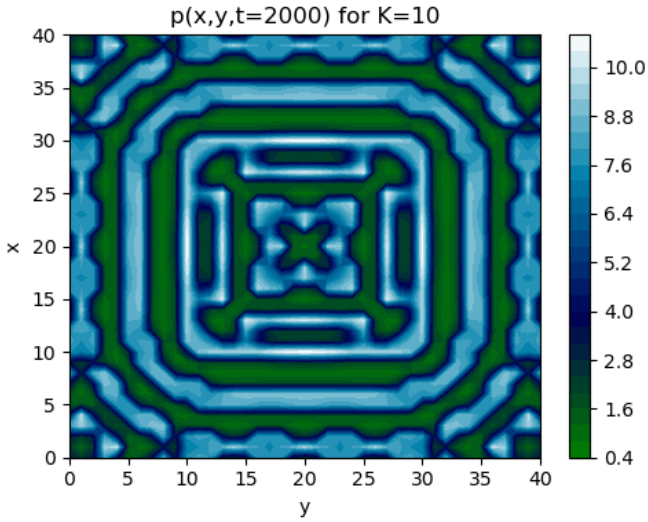
- $D_p = 1, D_q = 8, C = 4.5, dt = 0.01, N = 41$ and we vary the parameter K

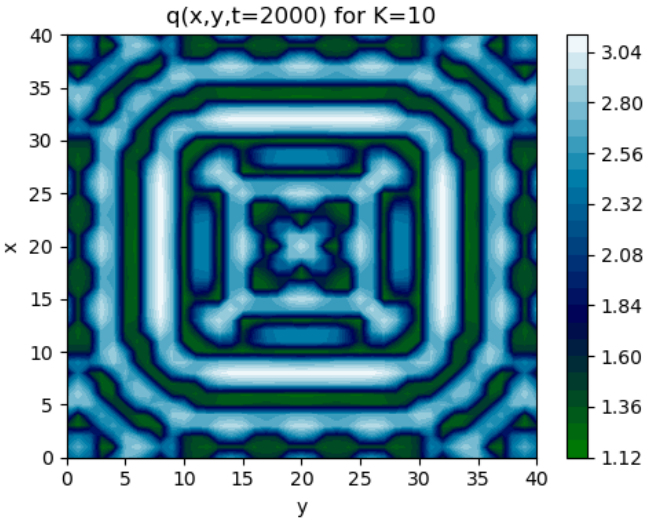


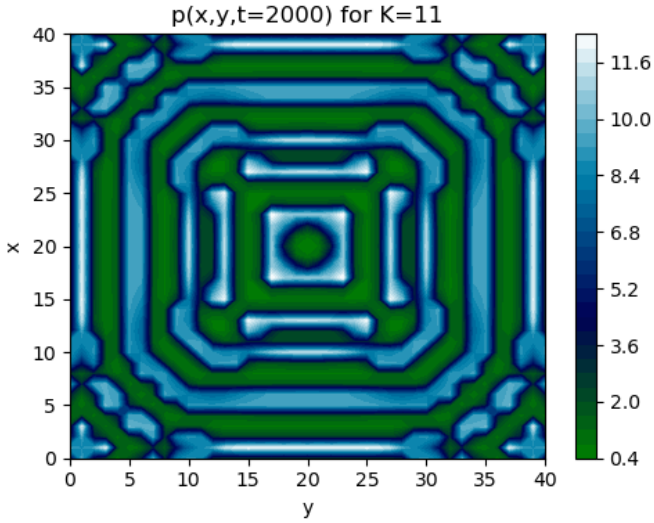


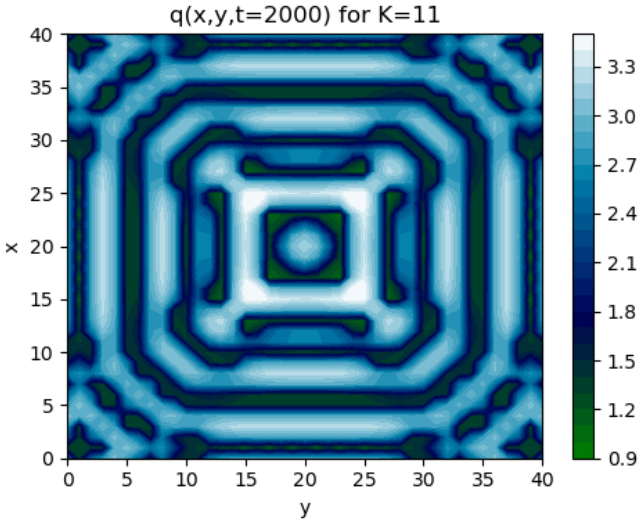

p(x,y,t=2000) for K=9
x
y
0 5 10 15 20 25 30 35 40
9.0 7.8 6.6 5.4 4.2 3.0 1.8 0.6

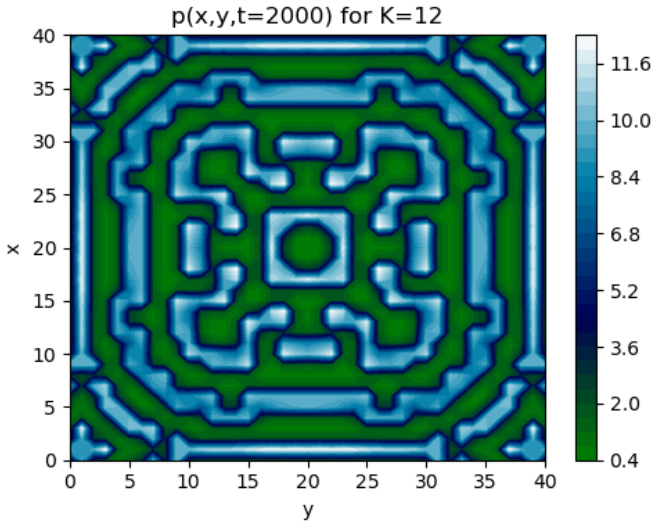

q(x,y,t=2000) for K=9
x
y
0 5 10 15 20 25 30 35 40
3.28 3.04 2.80 2.56 2.32 2.08 1.84 1.60 1.36 1.12

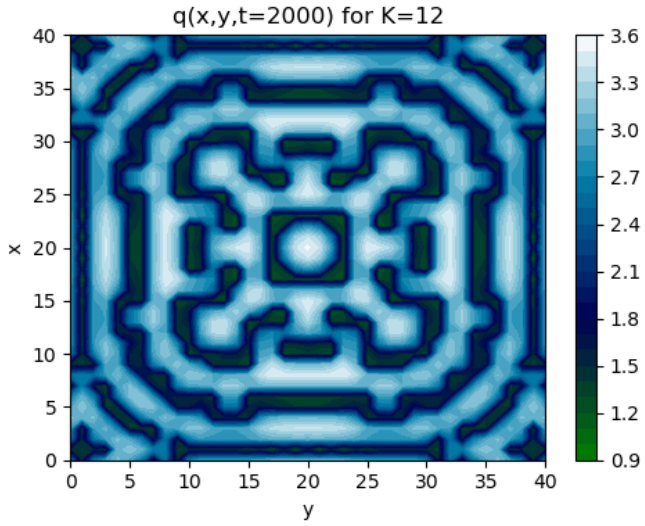

p(x,y,t=2000) for K=10
x
y
0 5 10 15 20 25 30 35 40
10.0 8.8 7.6 6.4 5.2 4.0 2.8 1.6 0.4


q(x,y,t=2000) for K=10
x
y
0 5 10 15 20 25 30 35 40
3.04 2.80 2.56 2.32 2.08 1.84 1.60 1.36 1.12


p(x,y,t=2000) for K=11
x
y
0 5 10 15 20 25 30 35 40
11.6 10.0 8.4 6.8 5.2 3.6 2.0 0.4


q(x,y,t=2000) for K=11
x
y
0 5 10 15 20 25 30 35 40
3.3 3.0 2.7 2.4 2.1 1.8 1.5 1.2 0.9





- $D_p = 1, D_q = 8, C = 4.5, dt = 0.01, K = 8$ and we vary the parameter N