Project 4 part 2

Niels August Davidsen (phx657)

Hand-in November 3rd. 2025

Reaction-Diffusion equations (PDEs)

To solve this assignment I wrote two functions for the Laplacian:

- The first used loops for constructing the correct boundary conditions for each site at a time. This was slow initially, but as it doesn't use any complicated *numpy* functions I was able to speed it up using *numba* making it a lot faster than even the vectorized version.
- The second used vectorized operations to help boost speed in the loop. This was way faster (about 10 times) as my first iteration using loops, but was outperformed by the *numba* optimized version.

The two algorithms can be seen in the picture below (fig. 1 or in the code file).

Otherwise I used a standard Forward Euler-method to solve the differential equations. I tried implementing the Crank-Nicholson-method, but had a lot of trouble constructing the Laplacian and using the correct operations for sparse matrices. I therefore stuck to the Euler-method.

I played around with Δt to find something that would give me stable solutions without costing too much in terms of speed. I ended up using $\Delta t = 0.01$. All systems were simulated to $t = 2000$ which means 2 million time steps. I used a system size $L = 40$ with the given initial conditions and used a grid-spacing $\Delta x = \Delta y = h = 1$.

```python
@njit
def laplacian(M, h=1.0):
    Lx, Ly = M.shape
    lap = np.zeros_like(M)
    for i in range(Lx):
        for j in range(Ly):
            up    = M[i-1, j] if i > 0 else M[i, j]
            down  = M[i+1, j] if i < Lx-1 else M[i, j]
            left  = M[i, j-1] if j > 0 else M[i, j]
            right = M[i, j+1] if j < Ly-1 else M[i, j]

            lap[i, j] = (up + down + left + right - 4 * M[i, j]) / (h * h)
    return lap


def laplacian_vec(M, h=1):
    M = np.pad(M, pad_width=1, mode='edge')
    up = M[:-2, 1:-1]
    down = M[2:, 1:-1]
    left = M[1:-1, :-2]
    right = M[1:-1, 2:]
    center = M[1:-1, 1:-1]
    lap = (up + down + left + right - 4*center) / h**2
    return lap

```

Figure 1: Laplacian functions used to solve the system of equations

First i tried simulation the system for $D_p = 1, D_q = 8, C = 4.5$ and for $K = 9$ and look at how it evolved over time. The first 3 parameters were kept constant, but K was later varied. The concentrations p and q are shown in the plots in fig. 2, where all plots look symmetric at all time steps, which is due to the Von Neumann boundary conditions.

I next ran the simulations again but this time for varying $K \in [7, 8, 9, 10, 11, 12]$. The plots fig. 3 show the result of all simulations for both concentrations q and p, but only shows the final time step of the simulation. All plots are again symmetric, but express different patterns according to their respective value K. Like before, it is noticeable, that the concentration of p generally has a much larger scale than q.

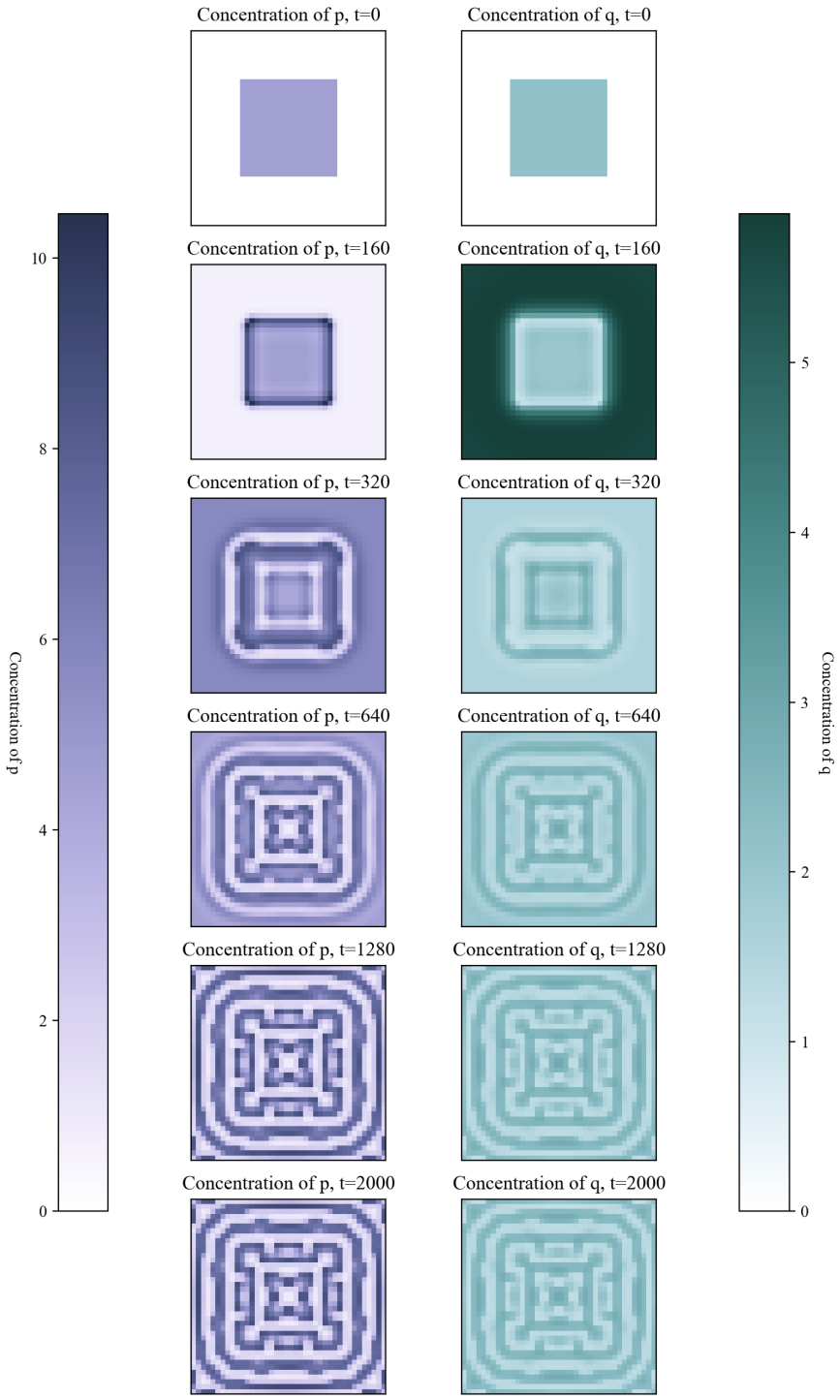

Figure 2: Concentration over time for a system with $K = 9$

Concentrations of p and q at $t = 2000$ for varying K

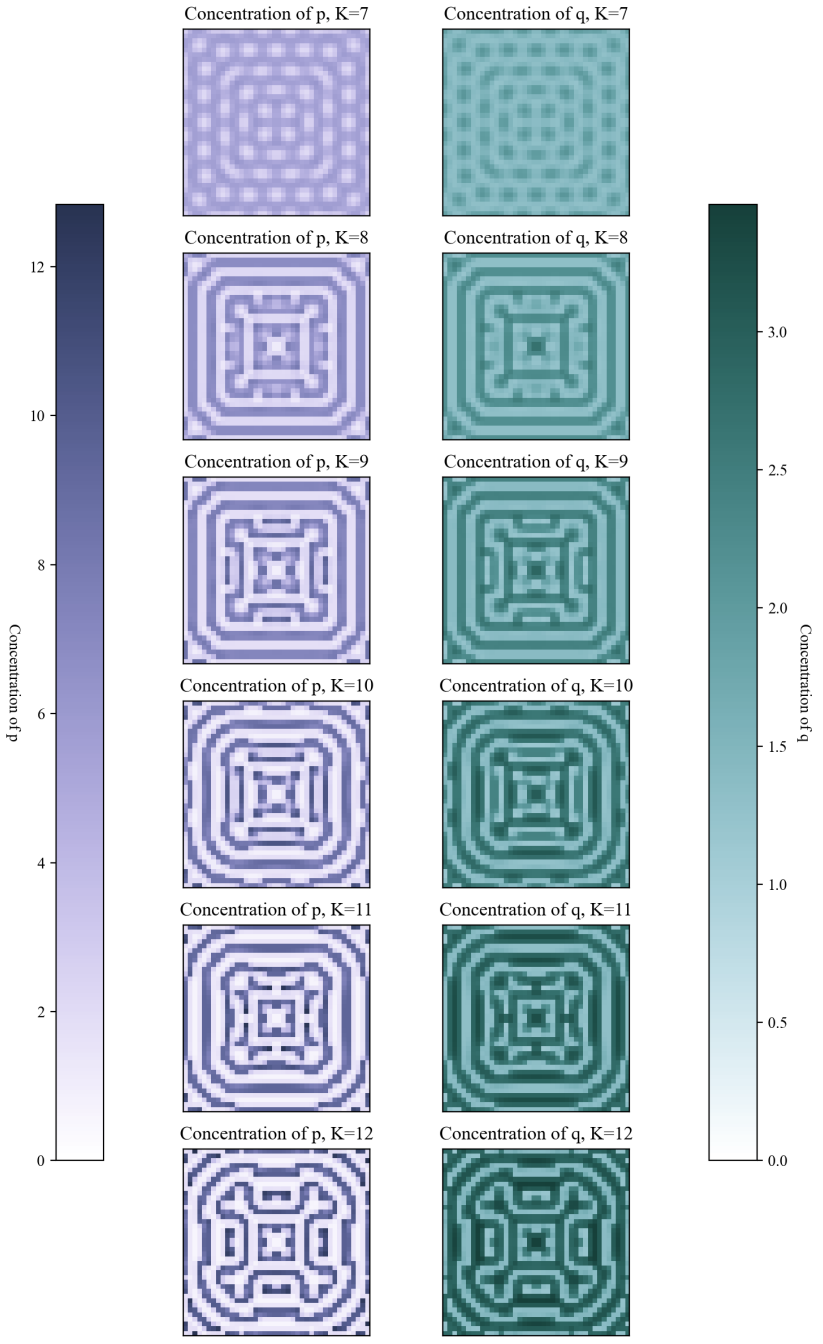

Figure 3: Simulations of systems with varying K from 7 to 12